TRUST BASED TIME-VARYING NETWORK TOPOLOGY FOR DISTRIBUTED CO-OPERATIVE CONTROL OF MULTI-CLASS MULTI-AGENT SYSTEMS

by

ANKUR VIPULKUMAR DALAL

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2018

To my family.

Acknowledgements

I would like to thank my supervising professor Dr. Frank L. Lewis for providing me with an opportunity to work in his lab at The University of Texas at Arlington Research Institute (UTARI). I am grateful for his continuous support and mentorship during my research.

I would like to acknowledge and thank my thesis advisor Dr. Kamesh Subbarao for all his guidance and support. I would also like to thank Dr. Robert L. Woods and Dr. Alan P. Bowling for taking interest in my research and serving on my thesis committee.

I would also like to extend my gratitude towards my colleagues Patrik Kolaric and Cody L. Lundberg for all the support that they have provided me while working at UTARI. I would also like to thank all the staff at UTARI for all their support.

April 12, 2018

Abstract

TRUST BASED TIME-VARYING NETWORK TOPOLOGY FOR DISTRIBUTED CO-OPERATIVE CONTROL OF MULTI-CLASS MULTI-AGENT SYSTEMS

Ankur Dalal, MS

The University of Texas at Arlington, 2018

Supervising Professor: Frank Lewis

With increased levels of autonomy in most of the engineering fields and booms in areas such as swarms, platoons and Internet of Things (IoT), communication and information flow has become a highly researched field. With advancements in autonomous vehicles (AVs) and drones in armed warfare, more and more focus is being laid on intercommunication between these vehicles and its surroundings as well as intra-communication among the fleets/swarms itself.

It is easier to deal with individual agents whereas it is quite challenging to deal with multi-agent systems especially with highly dynamic agents. In this thesis, we propose a general protocol for dealing with such multi-agent systems and how to manage dynamic agents. The approach is preliminarily based on graph theory for distributed multi-agent consensus control and contagion spread from

adversaries to the other agents is quarantined by methods of graph clustering. During the research, position consensus controller was experimentally verified and clustering methods were simulated on computer.

A major focus of the research is on how to accommodate for parting of existing adversaries from the group and allow for the entry of new agents to the flock as and when required in time. This aspect of the research allows for mitigating risk factors associated with hacked agents and couple new agents (with similar motives to that of the flock) to the flock.

Table of Contents

List of Illustrations

Introduction

1.1 Background and motivation

With increase in the levels of autonomy in almost every industry such as manufacturing, food processing, aviation and most recently transportation, there has emerged a need for connectedness. Internet of things (IoT) has been flourishing in the modern era and its basis lies in interconnection of devices and appliances over the internet. The network maybe a global one (like the internet) or a local one (like a Local Area Network), but the underlying motive remains the same, i.e. connectivity. Similar notion applies to robots as well. Whether they are autonomous ground vehicles for delivery or unmanned aerial vehicles in army, connectivity and interconnectedness play a crucial role in the performance of these vehicles, commonly known as “agents” in the control engineering society.

US Army invests huge sums in defense related research and development, a part of which comprises of research regarding multi-agent co-operative flights of unmanned aerial systems and its performance. With a revolutionary wave going across the automotive industry, all major automakers and tech giants like Google (division Waymo), Baidu and ride hailing service providers such as Uber and Lyft are investing heavily in the field of self-driving cars.

Fig. 1.1 Connected cars [34].

All of the above, in a way or other, are looking for interconnectivity for better performance and control. Not only tech companies, but companies like Facebook also make use of graph based analysis techniques on social networks for estimating group dynamics. With the above mentioned applications in mind and many more, the work presented in this thesis focusses on development and real-world simulation of strong network protocols for co-operative behavior and improvement of its performance.

In the light of above, efforts are made to develop and test position consensus controller with time-varying graph links. Most of the prior work conducted in this field lacks experimental verifications due to the complexity of the system. Different works propose different approaches such as differential

game theory [19], graph theory [20] and provide theoretical basis and computer simulation results. This thesis provides simulation results on existing theories with experimental verifications and simulations for the proposed way forwards.

1.2 Literature Review

One of the past works conducted in the field of consensus control of multi-UAV formation is [19] which formulates the formation control problem as a differential game problem. It considers fixed information graphs for exchange of information. The research in [19] proposes open-loop Nash strategy instead of classical Nash strategy. Aircraft dynamics are proposed as point-mass dynamics and high level formation control is designed. The models are cited from [21] [22] [23] which describes the point mass system as

$$\dot{x}_i = V_i \cos\gamma_i \cos\chi_i \tag{1}$$

$$\dot{y}_i = V_i \cos\gamma_i \sin\chi_i \tag{2}$$

$$\dot{h}_i = V \sin\gamma_i \tag{3}$$

$$\dot{V}_i = \frac{T_i - D_i}{m_i} - g \sin\gamma_i \tag{4}$$

$$\dot{\gamma}_i = \frac{L\cos\phi_i - m_i g \cos\gamma_i}{m_i V_i} \tag{5}$$

$$\dot{\chi}_i = \frac{L_i \sin\phi_i}{m_i V_i \cos\gamma_i} \tag{6}$$

With x_i as the displacement in downward direction, y_i as the cross-range displacement, h_i as the altitude, V_i as ground speed, γ_i is the flight path angle, χ_i is the heading angle, T_i is the engine thrust, D_i the drag and L_i the lift.

The research implements time-invariant directed graph for information flow. The game theory implementation seeks open loop Nash equilibrium for agents where the input U_i of agent i is a function of only the initial states

$z_1(0), \ldots, z_N(0)$ and t. The paper references to Pontryagin's minimum principle [24] for providing proof of optimality. The paper also mentions how open-loop Nash strategies are not applicable directly to directed graphs unless the graphs are fully connected. The author then proposes a sampled-Nash approach [25] which hints towards a possibility of handling time-varying information topology which might occur due to reasons such as communication failures and that is exactly what we are looking for and would build more on. The research presents a simulation result for five UAV system with some assumed parameters and constants and conclusion is derived on how distributed control over the multiple UAV system is achieved.

Other research focusing on a different concept of dynamic topology includes [26] which also considers the agent to be a point mass and considers switching graph topology for improvement in the performance of communication between agents. The research provides performance analysis for time varying topology in graph. One of the major assumption in the research is that the graph G at any time instant belongs to a finite set of graph structures with pre-defined number of nodes. The research work involves a typical motion equation as the system dynamics and control protocol is divided into two parts: a local feedback controller and a distributed state feedback controller. The end results provided involve computer based simulations which show the convergence of the control

protocol. This research also in part forms a basis for the research proposed in this thesis.

Another research on similar lines is [20] which also investigates the performance of consensus protocol based on time-varying graph topology and dynamic re-structuring of the links to sustain the co-operation among the agents under an external attack or internal performance degradation. The methodology adopted by the researchers involve a detection logic and internal connection restructuring which is essentially graph re-structuring. The theory involves indices which measure system performance. Thresholds are then chosen to classify the agents as either good or bad and based on that classification, the interconnection between agents is restructured. The way this is achieved is by checking for the indices at fixed intervals of time known as buffer time which is denoted by T_{buffer} . There is a performance index in terms of whether the agent is broken or not, corresponding to each agent. The index here is chosen as mean of the distance of the agent under inspection from its neighbors.

But according to the above, if the neighbors are under the influence of a wrong leader, the agent under consideration would be misguided and would sway away from the desired global consensus just like its neighbors. Hence, we put forward a concept of self-trust and consensus trust to account for personal performance as well as the opinion of neighbors. Not only this, but we also take into account the past history of every individual agent in order to minimize false

judging by neighbors. [20] provides a simulation result involving formation flight of 6 flights and compares the performance of controller with and without the faulty agent detection logic. The judging criteria and performance index calculation does not seem to be complete in this case and seem to be missing from considering important information. Therefore, we move to the next research which throws some light on what could possibly considered in order to correctly identify and classify agents into groups.

An approach to circumvent the above challenge could be to incorporate trust indices as a measure of sanity of an agent. In [27] a concept of trust establishment in autonomic networks was first introduced in [30] as a specific application of distributed trust management. Though the research mainly focusses on computer networks, the concept could be extended to any multi-agent network involving flow of information across nodes. Some other prior works focusing on trust in decentralized networks are [28] and [29]. An important concept in [27] is distrust which is essentially a negative trust index and is helpful in discarding or punishing the poor performing agents and neighbors.

Another interesting approach of segregating the agents into groups of good and bad agents is described in [31] where the inspiration to solve the network attack and prevention issue is addressed by methods of quarantining the affected nodes in order to minimize the spread of negative effects using graph topologies. It takes into consideration the extent to which a node and its neighbors are

affected and adopts a protocol of distributing the nodes into groups or districts which minimizes the spread. An effective way to do the partitioning of the topological graphs in order to achieve the desired grouping is required. The research in [32] provides a good insight by introducing graph clustering methods. The research illustrates multiple graph clustering methods such as normalized cuts, kernel k-means clustering and spectral clustering. The methods are shown to be quite useful on an undirected graph. Implementing it on a directed graph is quite a challenging task.

The simulations provided in the later chapters explain how the position consensus problem is simulated on a computer and control protocol involving graph theory is implemented. The simulations also incorporate trust calculations and edge weight modification accordingly. The classification and clustering problem is implemented in a novel way.

1.3 Challenges

Highly time-varying dynamical systems coupled with unpredictable behavior of agents pose a massive challenge in the maintenance of coordinated behavior of multi-agent systems. Co-operative control is highly researched and implemented area. Applications range from autonomous vehicle fleet to aerial swarms, network management to epidemic prevention and control, social network analysis and many others. But the major challenges pertaining to these applications and other include data lag, data loss, poor performing agents, handling of dynamic addition and parting of agents from the swarm and time-varying graphical structures.

Data lag depends on the network infrastructure and graph structures. Fully connected graphs help overcome this hurdle but has other issues and is rarely practically implementable due spatial spread of the network and power consumption. Therefore, strongly and weakly connected directed graphs are considered instead. Data loss may occur due to interference in network, external attack and hijacking.

Another major challenge to maintain good co-ordination amongst the agents is the stability of agents. There could be numerous reasons for this, not to mention the one's described in the prior paragraph. But a major cause of instability in agents is its dynamics. Agents under consideration are usually not perfect and have errors built into them. This makes it difficult to control

individual agent and multiple of these combined is an even worse case. Coupled with all these, the need of the hour is to add and part agents dynamically as time progresses. This is essentially helpful in maintaining continuum of highly mobile agents as they propagate through space and time. The problem arises due to the interaction of one graphical structure with a neighboring one, an isolated agent or even between agents and its surrounding. Most of the applications till date address the problem of finite time-invariant agents whereas this research proposes a novel concept to deal with time varying members.

1.4 Examples of agents

The graphs described in the previous chapters operate on some nodes or vertices. These nodes could be real physical system(s) or virtual centers of information exchange. Several systems could be modeled using this concept where some desired consensus is required among multiple of such nodes. It could be general behavior of quad-copters or reliable and secure data transfers. Some of them are explained below.

1.4.1 Vehicles as nodes

With modernization and boom in industrial automation, vehicle systems have seen some major applications of graph theory. Logistics is a huge, lucrative and continuously growing industry. This being one of the reasons for lots of recent R&D in this field.

Autonomous vehicles on streets is no longer a myth. It all began with the DARPA grand challenge back in March of 2004 [7] where multiple teams including teams from universities competed in an autonomous vehicle challenge. Today there are over a dozen companies [8] testing their autonomous vehicles on the streets of California. Freight platooning [9] another example of application of such autonomy. Each vehicle forming the platoon can be considered a node with one of them being the leader and rest being the followers. The vehicles adjust their speeds and try to maintain appropriate distances from each other. The aspect

that is of importance in maintaining such co-operative behavior is communication between the agents.

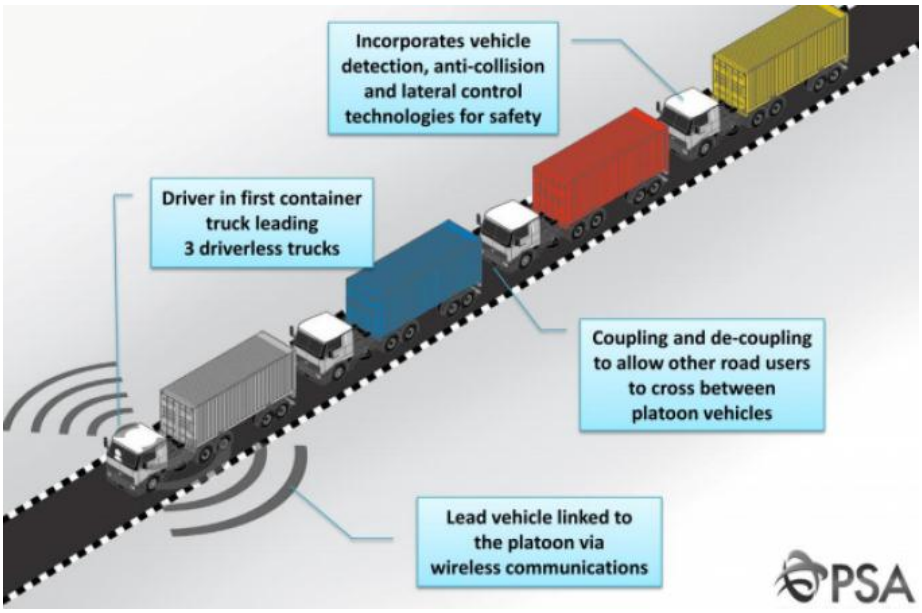

Fig. 1.2 Truck platooning [10].

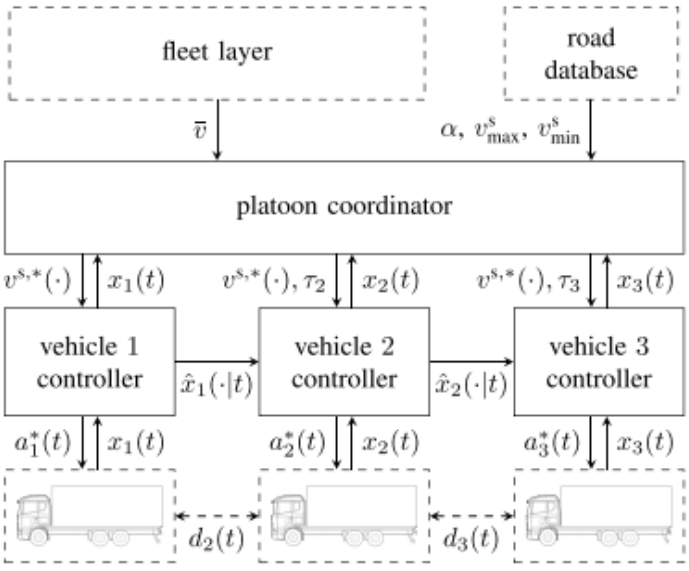

Fig. 1.3 Freight platooning [9]

1.4.2 Network nodes as agents

Similarly, network nodes that communicate to each other could be modelled as nodes of this graph and reliable communication across the network could be obtained.

1.5 Objectives and approach

1.5.1 Objectives

The main objective of this research is to develop a reliable infrastructure for obtaining consistent co-operative behavior. There are several methods of obtaining distributed control and achieving co-operative behavior but this research primarily focusses on graph theory based consensus control.

The existing consensus protocols involve static edge weights, directed or undirected graph structure, time-varying graph topologies and graph clustering. All these methods aim at improving the performance of the system. The method proposed in this research enhances the concept of trust propagation in graph and allows for dynamic edge weights which in turn allows for dynamic leader. The main idea behind this objective is to enable effective communication between nodes (vehicles in particular) and allow for co-operative behavior between existing agents and newly discovered agents in order to accommodate the new agents in the flock. This would help for smooth propagation of multiple flocks in a confined space. Similarly, there may exist situations where an agent might be required to leave the group voluntarily or under the influence of neighbors. Such scenarios occur when an agent is not performing well or has a different motive than the group.

List of publications

Accepted:

Kolaric, P., Chen, C., Dalal, A., & Lewis, F. (2018). Consensus controller for multi-UAV navigation. *Control theory and technology*.

Chapter 2

Mathematical preliminary and system description

2.1 Problem formulation

From various modules of chapter 1, it is necessary to reduce the problem to a much narrower definition in order to tackle the challenges that are listed. It may be noted that the ultimate goal of the research is to be able to develop a solution that could create a pipeline for the communication between autonomous vehicles in order to have successful operation without worrying about the individual case of interaction between vehicles or with its surroundings that may arise due to continuously varying spatial formation of the vehicle. But at educational and master's thesis level, it was not practically possible to develop the whole solution which could be used as an out-of-the box solution. Therefore, the research has focused on the development of general algorithm that could be applied not only to autonomous cars, but to any system where communication is a key to the operation and where formation control is desired. In order to extend the research to autonomous vehicles, significant efforts would be required.

We propose a position control problem involving n angents. The aim is to maintain a constant goal following in a particular fashion while maintaining the initial spatial formation. The algorithm must be robust enough to accommodate any changes in the number of agents as time progresses. This scenario may occur when some agents start to develop a consensus of their own and are influenced by

the wrong agents. This scenario would eventually lead to a similar behavior by other agents following the bad agent and the whole consensus would see high inputs in an undesired direction. In order to prevent such a scenario to build up, necessary preventive measures are proposed.

An index that would judge the performance of the agent according to its behavior with respect to time. The concept of trust is incorporated as seen in [27] but slightly differently in order to better criticize each agents. Two indices are introduced: self-trust and consensus trust. Both the indices evolve with time and consider the time history of agents while criticizing the agents. As the names suggest, self-trust is a measure of performance of an agent as seen by the agent itself while consensus trust is a performance index of an agent as seen by its neighbors. There may arise some doubts regarding this concept as to what would happen to the indices in case the critics (the neighbors) of a particular agent have poor performance (meaning a low self-trust). Obviously the critics in this case would be bad judges. These kind of circumstances are explained and dealt with in detail in later chapters.

Once the trust values for each of the agents is calculated, next we pose the problem of classifying them into adversaries or non-adversaries. We classify the non-co-operating agents as adversaries and the rest as non- adversaries based on the trust values. Upon successful classification of all the agents, we then move on to the problem where we need to minimize the spread of the contagion. We came

across several graph clustering techniques in [32] which explains algorithms for classifying nodes of an information graph into multiple classes based on some criteria. The methods suggested work best for undirected graph where the direction of flow of information is not taken into account but instead just represents a relation between nodes.

The thesis here aims to extend the graph clustering methods to directed graphs. But that alone would not suffice. Our ultimate objective is to add and part agents dynamically and in order to achieve that, we need a means to place graph cuts at the correct positions (graph weights) modifying the existing edge weights (increasing or reducing, but mostly reducing). This has to be done in such a fashion so as to reduce the spread of the negative effects of the adversarial agents to others and for that reason we incline to quarantine methods in [31].

2.2 Methods involved

2.2.1 Graph

Graph theory is a widely used theory in computer science, electrical engineering, biological studies [1], social network analysis [2] and several other fields. It also finds wide applications in shortest path search algorithms [3][4][5] like the Dijkstra's algorithm and A*.

Graph is a mathematical structure represented by a pair $G = (V, E)$ with $V = \{v_1, v_2, \dots, v_N\}$, where V are vertices and E are edges of the structure. $(v_i, v_j) \in E$ represents the edge from vertex v_i to the vertex v_j . If graph's edges have direction then graph is directed. If edges are not directed (which means that they are bidirectional), then the graph is said to be undirected. Visual example of a directed graph is shown on Fig. 1. Each edge has a weight that can model a strength of connection between the nodes. Number of edges that are entering a node is called in-degree, while number of edges that are exiting a node is called out-degree. Graph can be presented in a form of matrix where column index corresponds to a source of an edge and row index to a sink of an edge. Matrix element a_{ij} indicates weight of corresponding edge in the graph. Matrix constructed of elements a_{ij} is called Adjacency matrix (A). A is a very convenient as it enables analysis of graph theory in the field of linear algebra. $d_i = \sum_{i=1}^{n} a_{ij}$ is a row sum of A. d_i is again introduced because of its

convenience in future analysis. Matrix D is diagonal matrix of in-degrees, $= diag(d_i)$. Finally, we introduce Laplacian matrix $L = D - A$ which is important in analysis of graph's dynamical properties.

Types of graphs –

Graphs are primarily classified into directed graphs (or commonly called digraphs) and undirected graphs. The difference between the two as the name suggests is in the nature of the edge weights. If the edge weights are directional, the graph is said to be directed graph else it is called an undirected graph. Based on the direction of the edge weight, the edge might be an in-link into or an out-link from a node.

Various applications demand for different variants of these graphs. Graphs could be open ended or close ended, may or may not have spanning tree, could be fully connected or not and might or might not have self-links from nodes to itself.

A graph is said to be strongly connected if it has a spanning tree from every node, i.e., if it has a path from every node to every other. A graph could have all the edge weights equal or different weights.

There also exist special forms of graphs where the topology keeps switching with time [6]. Sometimes it is inherent in the system while at other times, it would be modeled in order to obtain better performances.

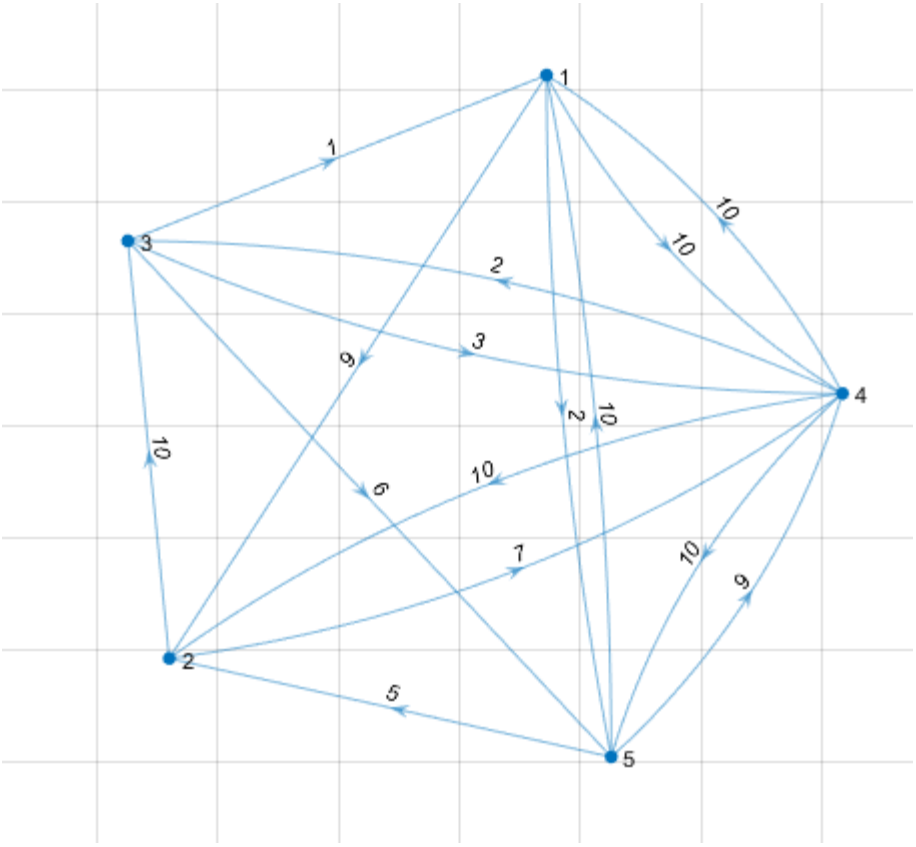

Fig. 2.1 Directed graph showing edge weights.

In the graph shown above, solid dots represent the vertices of the graph structure and the lines represent the edges and the numbers on these edges represent the edge weight. Agents under consideration are represented by the vertices and their mutual relationships are represented by the edge weights. The direction of the edge shows which vertex is influencer and which on is influenced.

2.2.2 Trust indices

Trust is a very general term. Some of the approaches like [27] and [30] have tried to define and quantify the word trust. Sometimes it is desired to have the value of trust as a binary value (like in [30]) and sometimes a floating point value (like in [27]). Sometimes a thresholding may be required while in other cases, thresholding might be applied prior to the calculation of trust index. So it generally varies from application to application. Another important formulization during implementation of trust indices is distrust index. Distrust is nothing but negative trust which could be used to heavily penalize disagreeing agents. But the idea behind all of them remain the same, i.e. to develop a measuring index in order to prioritize or weigh different agents/entities according to a given cost function.

In our case, the cost function would essentially be the positional tracking error. Based on how good the agent is following the leader, it would be assigned a trust index, better performance (low tracking error), higher trust value and vice-versa. For our discussion, we limit the trust values in the closed interval $[0,1]$ because negative trust values would mean driving away the adversaries (which could be done in extreme cases) and trust values higher than unity would mean changing of the leader (more on this in later chapters).

We incorporate two trust indices, namely self-trust index and consensus trust index. As the name suggests, the prior index takes into account the

performance of an agent as viewed by itself whereas the latter index is formed by the combined judgment of the neighbors of the agent. The main motive behind considering two indices instead of one is to reduce the number of outliers. The way this works is that each individual agent judges itself based on the inputs received and the output that it produced. This forms the basis for self-trust index. While consensus trust index is formed by the collective decision of the neighbors on how the agent under consideration performed and how it should have performed. It might be argued that consensus trust should suffice and there would not be any requirement of another index but self-trust helps weigh the neighbor judgment according to its own performance. For instance, an agent with multiple poor performing agents would be judged incorrectly as the agents are seeking different goals.

2.2.3 Clustering

After assigning trust indices to every agent, the next step is to classify the agents as adversaries and non-adversaries. Simple thresholding can be implemented to achieve the classification. Once the classification is complete, there may exist multiple groups, each segregated into either adversary or non-adversary. Depending on the group to which a particular agent belongs to, clusters could be formed around them to restrict the spread of contagion.

Now there comes an important decision making step, whether to let go of the bad agents or just reduce the dependency of good agents on adversaries. Some of the methods explained in [32] and [33] as shown below such as normalized cut, ratio cut and ratio association are some of the partitioning techniques that can be implemented.

Name	Objective Function
Ratio Association	$\underset{\mathcal{V}_1,\ldots,\mathcal{V}_k}{\text{maximize}} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c,\mathcal{V}_c)}{\lvert\mathcal{V}_c\rvert}$
Ratio Cut	$\underset{\mathcal{V}_1,\ldots,\mathcal{V}_k}{\text{minimize}} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c,\mathcal{V}\backslash\mathcal{V}_c)}{\lvert\mathcal{V}_c\rvert}$
Normalized Cut	$\underset{\mathcal{V}_1,\ldots,\mathcal{V}_k}{\text{minimize}} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c,\mathcal{V}\backslash\mathcal{V}_c)}{\text{degree}(\mathcal{V}_c)}$

Fig. 2.2 Examples of graph clustering objectives [33].

A concern here is the necessary trade-offs in order to maintain consensus in maximum number of agents while eliminating/reducing dependency on adversaries. The factors that would come into play while deciding the trades-offs consist of rate of information transfer which is basically dictated by the graph structure in our case. The information transfer rate determines how fast the contagion would spread and depending on that, we need to cluster the agents in such a way that we lose as minimum number of agents as possible.

Chapter 3

Analytical approach

3.1 Consensus controller

Consensus controller refers to a control protocol for maintaining co-ordination among agents in a system by distributed error minimization and control input. Efforts have been made to test the consensus algorithm by application of graph theory and improving its performance by implementing trust based time-varying graphical edge weights and dynamic leader and clustering of the existing graph into groups of adversaries and non-adversaries. For the purpose of demonstration, highly dynamically unstable system comprised of nano quad-copters called CrazyFlie 2.0 [11] are used.

The aim is to maintain coordinated flight of a swarm of UAV in a fixed formation inside a Vicon motion capture [12] tracking environment. The formation control involves maintaining fixed distances from neighbors and achieve an overall flight formation. Information is shared locally between agents, meaning only neighbors according to the graph are allowed to communicate with each other. This brings about the distributedness in the system.

3.2 Position consensus controller

Consider the following control protocol on a graph:

$$u_i(t) = \sum_{j \epsilon N_i} a_{ij}(x_j(t) - x_i(t)) \tag{7}$$

Where $x_i(t)$ is the current state of the node i and u_i is the control input generated by consensus protocol. Graph states x_i can be controlled by applying the following control law:

$$\dot{x}_i(t) = u_i(t) \tag{8}$$

Equations of closed loop dynamics can then be rearranged:

$$\begin{aligned} \dot{x}_i(t) &= -x_i \sum_{j \in N_i} a_{ij} + \sum_{j \in N_i} a_{ij} x_j \\ &= -d_i x_i + [a_{i1} \dots a_{iN}][x_1 \dots x_N]^T \end{aligned} \tag{9}$$

Vectorization of x_i into $x = [x_1 \dots x_N]^T$ gives more compact way to rewrite Eq. 18 as:

$$\dot{x} = (A - D)x = -Lx \tag{10}$$

Where L represents the Laplacian matrix.

We introduce Δ_i of UAV i from the center of the formation. Different combinations of Δ_i can now be used to encode various formations. x_0 is the agent leader and is directly linked to those agents that have non-zero pinning gain g_i. The reference is generated by the leader and is effectively transmitted to the whole system through the distributed communication network.

The neighborhood error is defined as

$$e_i = \sum_{j \epsilon N_i} a_{ij}\left(x_j - \Delta_j - x_i + \Delta_i\right) + g_i(x_0 - x_i + \Delta_i) \tag{11}$$

And neighborhood time derivative of the error

$$\dot{e}_i = \sum_{j \,\epsilon\, N_i} a_{ij}\left(\dot{x}_j - \dot{\Delta}_j - \dot{x}_i + \dot{\Delta}_i\right) + g_i\left(\dot{x}_0 - \dot{x}_i + \dot{\Delta}_j\right) \tag{12}$$

If we assume that the position of the leader (x_0) and the relative positions between the agents ($\Delta_i, \forall i$) are not changing during the experiment then

$$\dot{e}_i = \sum_{j \,\epsilon\, N_i} a_{ij}\left(\dot{x}_j - \dot{x}_i\right) + g_i(-\dot{x}_i) \tag{13}$$

The alias of ideal positions displaced by Δ_i is introduced

$$x_i^a = x_i - \Delta_i \tag{14}$$

For $i = 1,2, \ldots, N$. From above equation, position errors between the two agents are given as

$$\delta_{ij} = x_j^a - x_i^a \tag{15}$$

From the above equation, the neighborhood error defined previously now has the form

$$e_i = \sum_{j \epsilon N_i} a_{ij}\left(x_j^a - x_i^a\right) + g_i(x_0 - x_i^a) \tag{16}$$

$$\dot{e}_i = \sum_{j \in N_i} a_{ij}(\dot{x}_j - \dot{x}_i) + g_i(-\dot{x}_i) \qquad (17)$$

3.3 Convergence of the errors

Define $1_N = [1, 1, \dots, 1]^T\ R^N$ with all N elements ones. The global forms of (25) and (26) are now expressed as

$$e = -\big((L+G)\ \otimes I_n\big)(x^a - 1_N \otimes x_o) \tag{18}$$

$$\dot{e} = -\big((L+G) \otimes I_n\big)\dot{x} \tag{19}$$

In order to further develop the discussion on that hypothesis, the second order dynamics is first calculated

$$\ddot{e} = -\big((L+G) \otimes I_n\big)\ddot{x} \tag{20}$$

Plugging (19) in (29) gives

$$\begin{aligned} \ddot{e} = & -\big((L+G) \otimes I_n\big)[(I_n \otimes A)x + (I_n \otimes B)v\,] \\ & = -((L+G) \otimes I_n)(A_g x + B_g v) \end{aligned} \tag{21}$$

Where $A_g = I_N \otimes A$ and $B_g = I_N \otimes B$. Without the loss of generality, further simplification is introduced

$$\ddot{e} = -\big((L+G) \otimes I_n\big)\big(A_g x + u\big) \tag{22}$$

Where $u = B_g v$ is defined as the global vector of inputs.

To drive both position and velocity to zero, we use the sliding mode control and define the sliding mode error as

$$r = \dot{e} + \Lambda e \tag{23}$$

Where Λ is a positive definite. e is bounded as long as r is bounded let $\lambda = diag(\lambda_i)$ be N dimensional diagonal matrix with λ_i on the diagonal, each corresponding to one agent. Then $\Lambda = \lambda \otimes I_n$. Taking the time to derivative of (32) yields

$$\dot{r} = \ddot{e} + \Lambda\dot{e} = -\big((L+G) \otimes I_n\big)\big(A_g x + u\big) \\ -\Lambda\big((L+G) \otimes I_n\big)\dot{x} \tag{24}$$

By using Kronecker rule $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$ (under the assumption that dimensions of A, B, C, D allow multiplications AC and BD), we introduce the following for convenience

$$Z = \big((L+G) \otimes I_n\big) \\ Z^A = \big((L+G) \otimes I_n\big)A_g = \big((L+G) \otimes A\big) \tag{25} \\ Z_\lambda = (\lambda \otimes I_n)\big((L+G) \otimes I_n\big) = \Big(\big(\lambda\,(L+G)\big) \otimes I_n\Big)$$

Rewrite (33) as

$$\dot{r} = -Z^A x - Z_\lambda \dot{x} - Zu \tag{26}$$

Based on the undirected graph topology, we make some assumptions useful for the control design.

Assumption 1

L is irreducibly diagonally dominant matrix.

Assumption 2

G has at least one diagonal entry.

It is not restrictive to make assumptions 1 and 2, since many practical multi-agent systems fall under that category.

The control law locally and globally is introduced as

$$u_i = u_{1i} + K_i r_i \tag{27}$$

$$u = u_1 + (I_N \otimes K_i)r = u_i + Kr \tag{28}$$

Where matrix $K_i = K_j, \forall i, j$ is used for control design. Assume that u_i is specified later in the theorem. Under that assumption, re-writing (28) in the global form

$$\dot{r} = -\big((L+G) \otimes K_i\big)r = -ZKr \tag{29}$$

Lemma 1

Let assumptions 1 and 2 hold. If we define

$$W = diag(w_i) = diag(\frac{1}{q_i}) \tag{30}$$

Where

$$q = (L+G)^{-1}1_N \tag{31}$$

Then, both $W\ and\ Q = W(L+G) + (L+G)^T W$ are positive definite. Additionally, if Q is a positive definite then $Q \otimes I_n$ is also positive definite. $Q = W(L+G) + (L+G)^T W$

Proof: the first assertion follows the results in [35]. For the second assertion, we obtain that eigenvalues of a matrix generated by Kronecker product are cross-products of all possible combinations of eigenvalues from matrix Q and

I_n. Both Q and I_n are positive definte and corresponding eigenvalues all positive. This completes the proof.

Assumption 3

Let $\bar{x}$ be the measurements of ideal state vector x

$$\begin{aligned} \bar{x} &= x + \mu_1 \\ \dot{\bar{x}} &= \dot{x} + \mu_2 \\ \bar{x} &= x_0 + \mu_0 \end{aligned} \tag{32}$$

Then define the bound on sensor noise

$$\begin{aligned} ||\mu_1|| &< \mu_{1,B} \\ ||\mu_2|| &< \mu_{2,B} \\ ||\mu_0|| &< \mu_{0,B} \end{aligned} \tag{33}$$

Where $\mu_{0,B}$ denotes the measurement error of leader's position.

Sliding mode error injected with noises is

$$\bar{r_i} = \dot{\bar{e}}_i + \lambda_i \bar{e}_i \tag{34}$$

Where

$$\bar{e_i} = \sum_{j \in N_i} a_{ij}\left(\bar{x}_j - \Delta_j - \bar{x}_i + \Delta_i\right) + g_i(\bar{x}_0 - \bar{x}_i - \Delta_i) \tag{35}$$

$$\bar{e_i} = \sum_{j \in N_i} a_{ij}\left(\dot{\bar{x}}_j - \dot{\bar{x}}_i\right) + g_i(-\dot{\bar{x}}_i) \tag{36}$$

The global form of noisy error is

$$\bar{r} = \dot{\bar{e}} + \Lambda\bar{e} \tag{37}$$

Where

$$\bar{e} = -\big((L+G) \otimes I_n\big)(\bar{x}^a - 1_N \otimes \bar{x}_0)$$

$$= -\big((L+G) \otimes I_n\big)(x^a + \mu_1 - 1_N \otimes x_0 - 1_N \otimes \mu_0) \tag{38}$$

$$= -\big((L+G) \otimes I_n\big)(\mu_1 - 1_N \otimes \mu_0) + e$$

$$= -Z\mu_2 + \dot{e}$$

$$\dot{\bar{e}} = -Z\mu_2 + \dot{e} \tag{39}$$

$$\ddot{\bar{e}} = -\big(A_g\bar{x} + u\big) \tag{40}$$

$$= -ZA_g\mu_1 + \ddot{e}$$

Extracting the ideal sliding mode error from the measured error gives

$$\bar{r} = r - \Lambda Z(\mu_1 - 1_N \otimes \mu_0) - Z\mu_2 \tag{41}$$

Substituting (42) and (41) into (26) yields

$$\dot{\bar{r}} = \dot{r} - ZA_g\mu_1 - \Lambda Z\mu_2 \tag{42}$$

Definition 1

The signal z(t) is said to be uniformly ultimately bounded (UUB) with the ultimate bound *b*, if given positive constants *b* and *c* for every $d \in (0, c)$, there exists $T(d, b)$, independent of t_0, such that

$$||z(t_0)|| \leq d \Rightarrow ||z(t)|| \leq b, \forall t \geq t_0 + T \tag{43}$$

Theorem 1

Let assumptions 1-3 hold. Defining the sliding mode error dynamics as (26). Select the control policy for the local agent as

$$u_i = K_i \bar{r}_i - A\bar{x}_i - \lambda_i \dot{\bar{x}}_i \quad (44)$$

Assume that $\lambda_i = \lambda_j, \ \forall i, j$ (53). Consider the error dynamics (38) and design matrices *Q* and *R*. Pick the following control gain

$$K_i = R^{-1}P \quad (45)$$

$$K = (I_N \otimes K_i) \quad (46)$$

Where *P* is the unique positive definite solution of control algebraic Riccati equation

$$0 = Q - PR^{-1}P \quad (47)$$

The control law (53) with the gain (54) guarantees asymptotic stability for (38). Moreover it stabilizes the system (19) and makes the ideal tracking error (20) UUB.

The controller drags all aliases x_i^a for $i = 1, \dots, N$ to the center of the formation. If all aliases are in the center of the formation, no action from the controller is required. Neighborhood errors quantify the spatial disharmony of the formation. The controller determines an action u_i of the agent i such that it recovers consensus. Λ is a tuning parameter. As given, controller resembles well known standard PD controller.

3.4 Simulation

3.4.1 System model

In order to verify the validity of the above proposed consensus controller, a computer simulation is carried out on Matlab. Initially, five agents are simulated. An input is provided to the agent labelled as 1 . This agent acts as the leader. There are four other follower agents labeled $2\ to\ 5$. As the control protocol, following equations is used

$$u_i = K_c \left[\sum_{j \in N_i} a_{ij}\left(x_j - \Delta_j - x_i + \Delta_i\right) + g_i(x_0 - x_i + \Delta_i) \right] \quad (48)$$

$$G = \begin{bmatrix} a_{11} & \cdots & a_{1j} \\ \vdots & \ddots & \vdots \\ a_{i1} & \cdots & a_{ij} \end{bmatrix} \quad (49)$$

The dynamics of the system are kept simple for simulation purposes and is modelled as

$$pos = \begin{bmatrix} x \\ y \end{bmatrix} \quad (50)$$

$$x_{t+1} = (x_t + u_x) + \eta_w \ \forall i \quad (51)$$

$$y_{t+1} = \left(y_t + u_y\right) + \eta_w \ \forall i \quad (52)$$

To add a dimension of reality to the simulation, the agents are infused with Gaussian white noise denoted by η_w.

3.4.2 Trust model

The trust function in the system is modelled to account for the past history of error and current error of the agent and modifies the edge weight associated with that particular agent accordingly. An important point to note here is that depending on the performance of the agent, other agents may lose dependency on a bad agent while the bad agent still maintaining similar dependence on the other agents. Before we can formulate trust index, we need to formulate error indices as explained below.

We define a past error history E_p^s as,

$$E_p^s = \int_0^t (x_0 - (x_i - \Delta_i)) \tag{53}$$

And current error as E_c^s as,

$$E_c^s = (x_0 - (x_i - \Delta_i)) \tag{54}$$

We also define a weighting matrix W (belongs to R 1x2) as,

$$W^s = [w_1^s \quad w_2^s] \tag{55}$$

where w_1^s represents the weight factor for past errors and w_2^s represents the weight factor for present error and the factors must satisfy the condition

$$w_1^s + w_2^s = 1 \tag{56}$$

in order to preserve the total error. We then define error index δ^s as

$$\delta^s = [w_1^s \quad w_2^s] \begin{bmatrix} E_p^s \\ E_c^s \end{bmatrix} \tag{57}$$

And we define self-trust index τ_s as

$$\tau_s = \frac{(\delta^s - \delta_u)^2}{((\delta_u - \delta_l)^2)} \quad (58)$$

3.4.3 Simulation cases

Case 1 – Fully connected graph.

Initial displacements of the agents are as follows

$$pos = (x, y) = \begin{bmatrix} 50 & 60 \\ 10 & 10 \\ 20 & 20 \\ 80 & 20 \\ 90 & 10 \end{bmatrix} \quad (59)$$

$$G = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} \quad (60)$$

The agent 1 is modelled to be the leader as

$$G_l = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (61)$$

The leader is modelled with an agent input of

$$u_1 = [0\ 0.6] \quad (62)$$

Result:

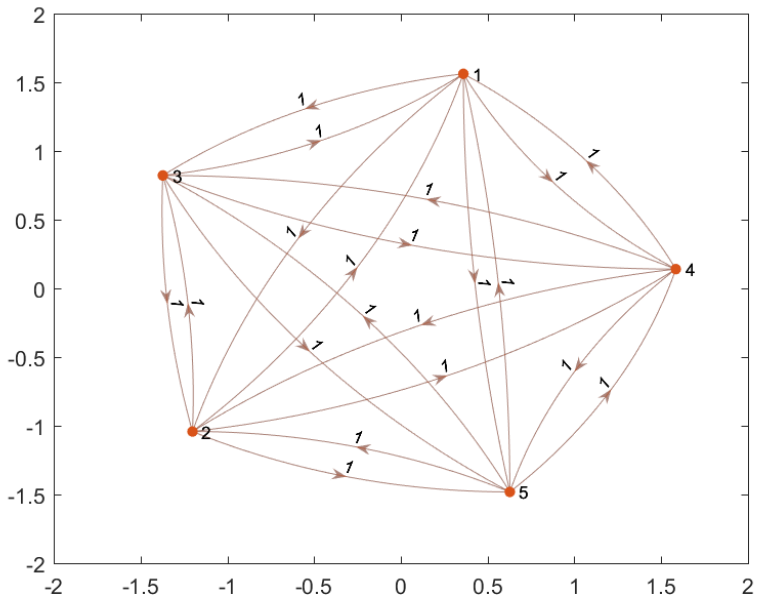

Fig. 3.1 Graph showing the edge weights.

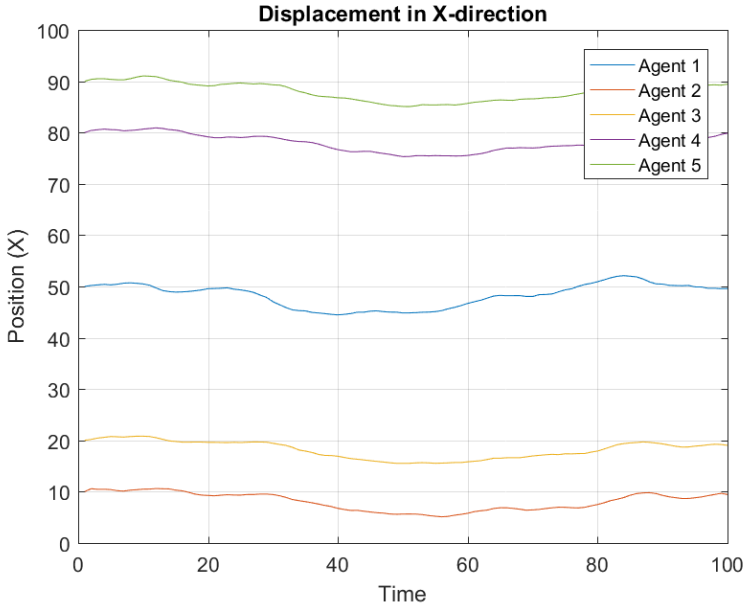

Fig. 3.2 Displacement in X-direction.

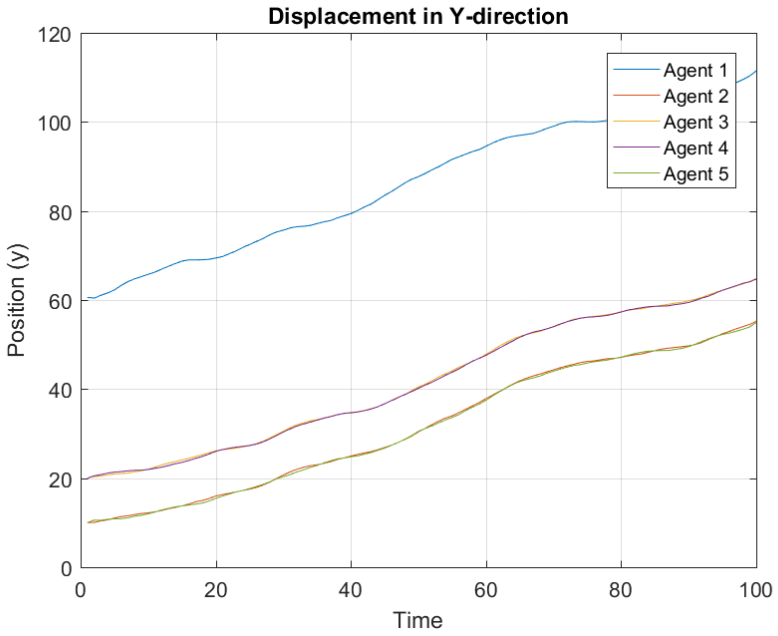

Fig. 3.3 Displacement in Y-direction.

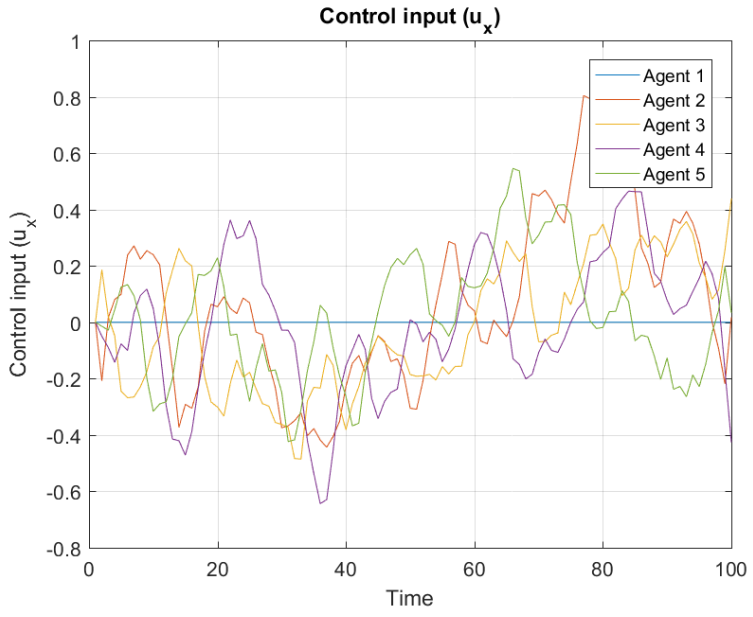

Fig. 3.4 Control input u_x.

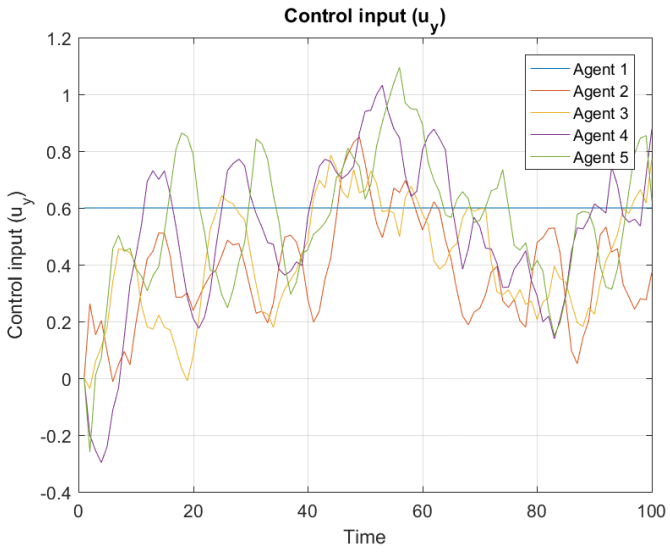

Fig. 3.5 Control input u_y.

Conclusion:

From the above, it is concluded that the agents are trying to follow the leader. It can be seen in the control input plots how agents try to keep up with each other in order to maintain the formation.

Case 2 – Random graph with trust implementation.

Initial displacements of the agents are as follows

$$pos = (x, y) = \begin{bmatrix} 50 & 60 \\ 10 & 10 \\ 20 & 20 \\ 80 & 20 \\ 90 & 10 \end{bmatrix} \quad (63)$$

$$G = \begin{bmatrix} 0 & 1 & 1 & 1 & 4 \\ 5 & 0 & 5 & 3 & 1 \\ 1 & 3 & 0 & 5 & 5 \\ 5 & 5 & 3 & 0 & 5 \\ 4 & 5 & 5 & 5 & 0 \end{bmatrix} \quad (64)$$

The agent 1 is modelled to be the leader as

$$G_l = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (65)$$

he leader is modelled with an agent input of

$$u_1 = [0\ 0.6] \quad (66)$$

Result:

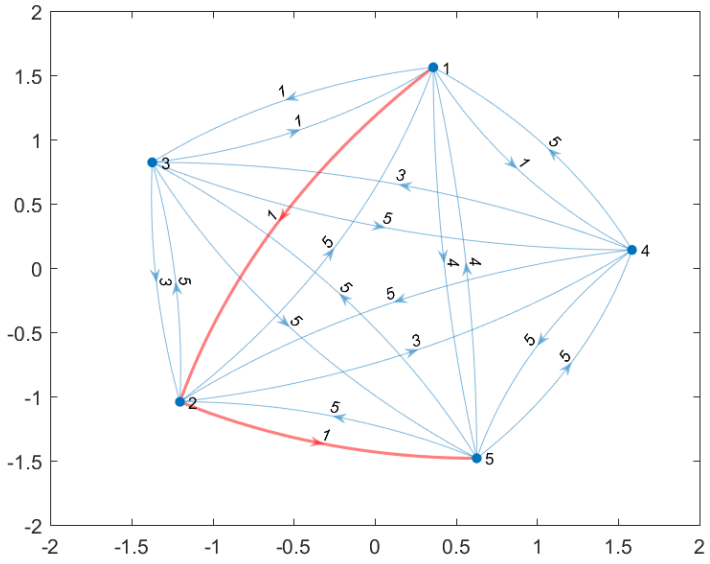

Fig. 3.6 Graph showing the edge weights.

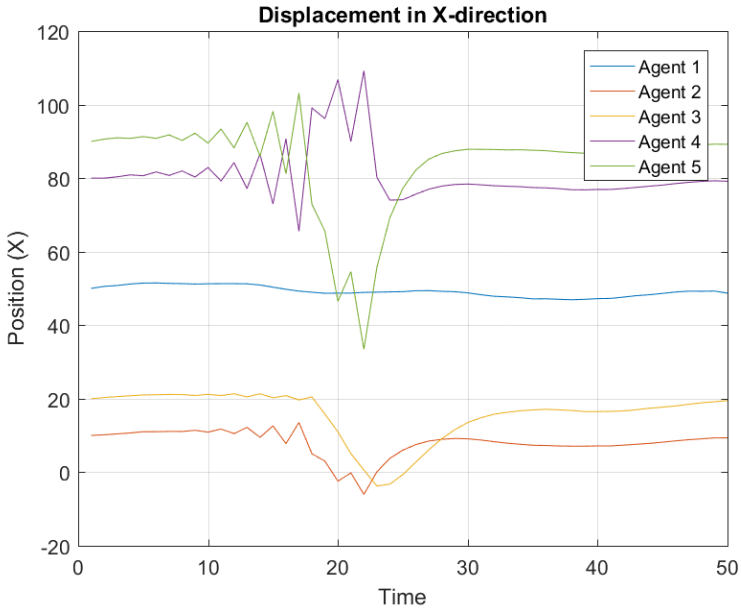

Fig. 3.7 Displacement in X-direction.

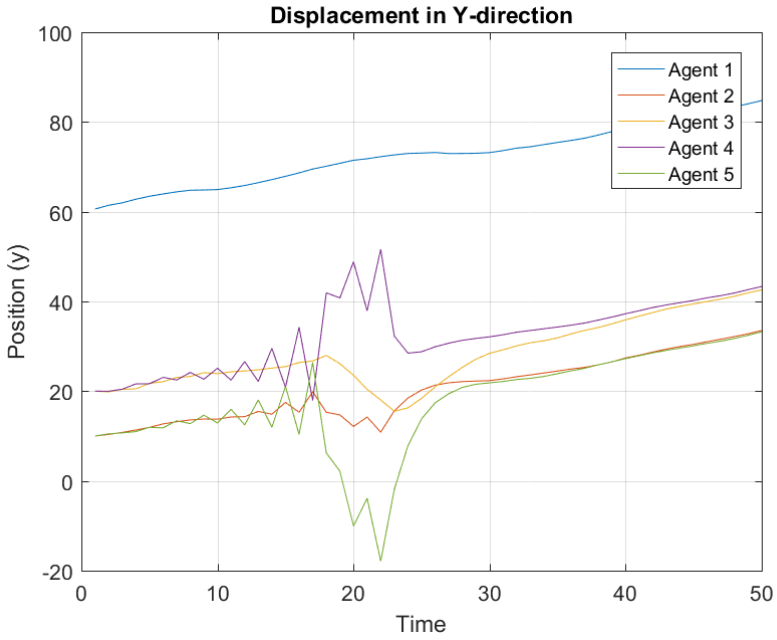

Fig. 3.8 Displacement in Y-direction.

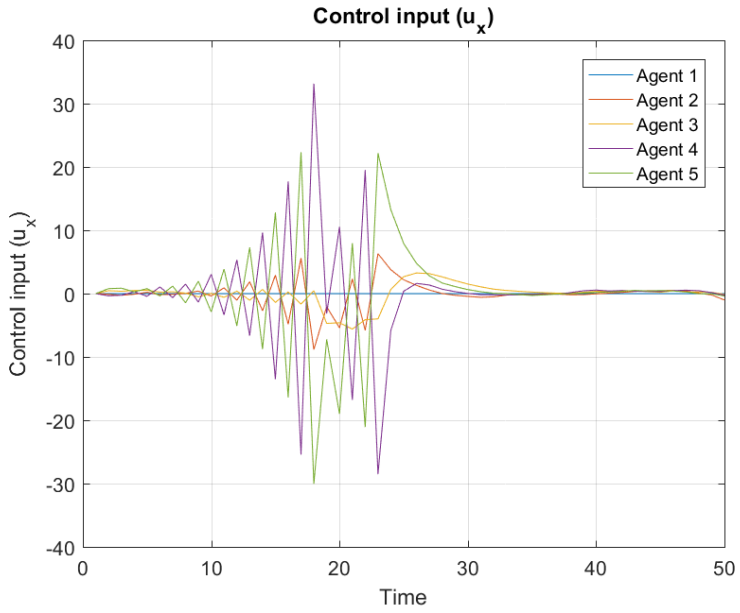

Fig. 3.9 Control input u_x.

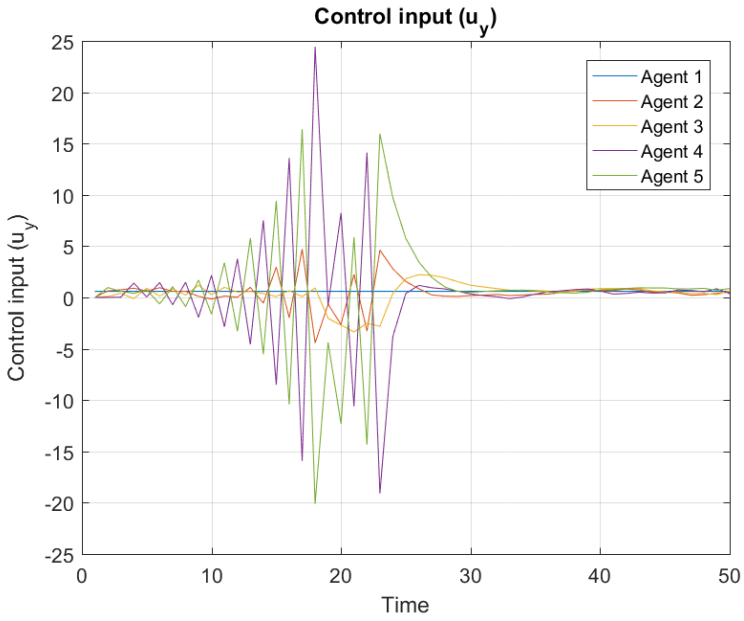

Fig. 3.10 Control input u_y.

Conclusion:

From the above, it is evident that due to unsymmetrical and random graph structure, the controller initially fails to respond to the position formation objectives and this is quite the scenario in real world situations. But we see that as trust kicks in at $t = 20$, the errors converge and formation is regained.

Chapter 4

Position consensus controller experiment

4.1 Experimental setup and system architecture

4.1.1 Crazyflie 2.0

The CrazyFlie 2.0 is a nano quadcopter weighing only 27 grams. It is an open source product developed by bitcraze.io for research and development purpose. Crazyflie 2.0 has two options for wireless connectivity: Bluetooth Low Energy (BLE) and radio communication through Crazyradio PA. The quadcopter is powered through an onboard battery that can be recharged using a standard micro USB cable. Each charge can provide about 7 minutes of flight time. This factor makes it challenging to use multiple of these for formation flight because it is a known fact that with change in charge levels, the dynamics of quadcopter change which also makes it an ideal system to test the performance of consensus algorithms developed by us. It supports wireless firmware update as well. All the parts could be quickly assembled and the quad-copter is designed to break at cheap replaceable plastic parts in case of crash and the parts are available as spares on the company's website. Also, due to the light weight characteristic of the aerial vehicle, the crash is not too destructive due to low kinetic energy during collision.

Fig. 4.1 CrazyFlie 2.0 with Vicon tracker markers.

Some of the other hardware specifications are as follows [14]:

- Mechanical specification
    - Weight: 27g
    - Size (WxHxD): 92x92x29mm (motor-to-motor and including motor mount feet)
- Radio specification
    - 20 dBm radio amplifier tested to > 1 km range LOS with Crazyradio PA
    - Bluetooth Low Energy support with iOS and Android clients available (tested on iOS 7.1+ and Android 4.4+)
    - Radio backwards compatible with original Crazyflie and Crazyradio
- Micro-controllers

        - STM32F405 main application MCU (Cortex-M4, 168MHz, 192kb SRAM, 1Mb flash)
        - nRF51822 radio and power management MCU (Cortex-M0, 32Mhz, 16kb SRAM, 128kb flash)
- μUSB connector
    - On-board LiPo charger with 100mA, 500mA and 980mA modes available
    - Full speed USB device interface
    - Partial USB OTG capability (USB OTG present but no 5V output)
- IMU
    - 3 axis gyro (MPU-9250)
    - 3 axis accelerometer (MPU-9250)
    - 3 axis magnetometer (MPU-9250)
    - High precision pressure sensor (LPS25H)
- Flight specification
    - Flight time with stock battery: 7 minutes
    - Chraging time with stock battery: 40 minutes
    - Max recommended payload weight: 15 g
- Expansion connector with
    - VCC (3.0V, max 100mA)
    - GND

- VCOM (unregulated VBAT or VUSB, max 1A)
- VUSB (both for input and output)
- I2C (400kHz)
- SPI
- 2 x UART
- 4 x GPIO/CS for SPI
- 1-wire bus for expansion identification
- 2 x GPIO connected to nRF51

- 8KB EEPROM

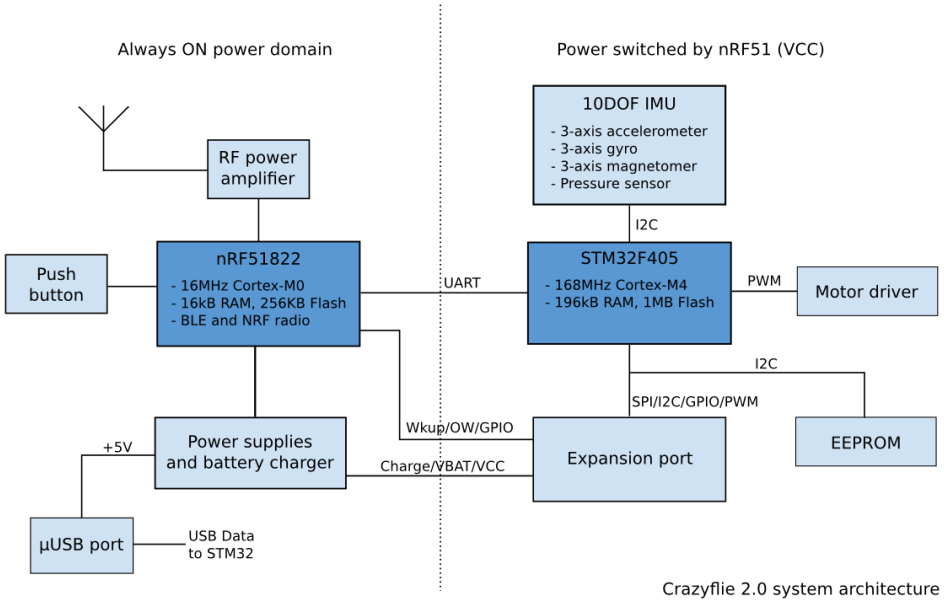

Fig. 4.2 CrazyFlie 2.0 system architecture [15].

Each system consists of nRF51822 and STM32F405. nRF51822 which has ARM Cortex-M0 serves for radio functionality and responsible for power management but it is not powerful enough to do the other controller computations as required for the operation of CrazyFlie. It uses CRTP (Crazy Real-Time Protocol) and BLE for radio communication. The nRF51 chip was designed to run off of a coin battery and hence is highly power efficient.

The other part is STM32F405 which is used for computation and processing of GPS and IMU data and other computationally heavy algorithms. This part has sufficient memory to accept new code as well as log sensor data. All other flight control such as motor control and telemetry is also handled by STM32F405. There is inter communication link setup between STM32F405 and nRF51822.

4.1.2 CRTP

CRTP stands for Crazy Real-Time Protocol. This protocol is used to communicate with the CrazyFlie wirelessly via CrazyRadio or CrazyRadio PA [16].

Fig. 4.3 CrazyRadio PA [16].

4.1.3 Vicon tracker

Vicon Tracker is a motion capture system which implements localization by principle of reflection of light. It involves infrared cameras to track specially designed highly reflective markers and provides the location in real-time [17]. Multiple of such infrared cameras could be used in conjunction to provide better pose estimates. For the purpose of our experiments, 16 such cameras were employed. The raw data was processed on a Windows 7 machine using the Tracker software. The software is optimized to track multiple objects in real-time [18]. The processed data output is in the form of position and orientation of the object as created in the software. This data is broadcasted via a router and made available to other users through ROS channel under a specific topic name.

4.2 Flight tests

The experimental test flights are carried out with a swarm of 3 CrazyFlie 2.0s. The take-off sequence takes off individual quad-copter at predefined time intervals. Once all the quad-copters achieve an initial goal formation location, the position controller switches from goal following controller to consensus controller. The positions and orientations are in the form of a vector

$$\begin{bmatrix} x \\ y \\ z \\ R \\ P \\ Y \end{bmatrix}$$

Where (x, y, z) represent the position in three directions and (R, P, Y) denote the roll, pitch and yaw[1] respectively. All these values are coming from the Vicon tracker and are therefore in a global world frame. In order to perform local neighborhood error calculations, these need to be transformed to local agent frames. This is taken care by using pre-built ROS functions.

A total of 3 test flights are conducted, each with different graph topology as well as leader weights.

The initial displacement vectors of the agents for all of the three flights are taken as (in the order of X,Y,Z)

[1] *Note: Quaternions are used in place of RPY in code in order to avoid singularities.*

$$agent\ 1: [\ 0.5, 0.5, 0.0\ ]$$

$$agent\ 2: [\ -0.5, -0.5, 0.0\ ]$$

$$agent\ 3: [\ -0.5, 0.5, 0.0\ ]$$

(All link lengths are in meters.)

4.2.1 Test flight 1

For the first test flight, the graph topology is

$$G = \begin{bmatrix} 0 & 3 & 0 \\ 3 & 0 & 2 \\ 0 & 2 & 0 \end{bmatrix}$$

Agent 1 is connected to the leader and other two agents follow agent 1 as per the graph topology.

The position variation of the quad-copters in X and Y directions[2] are shown below

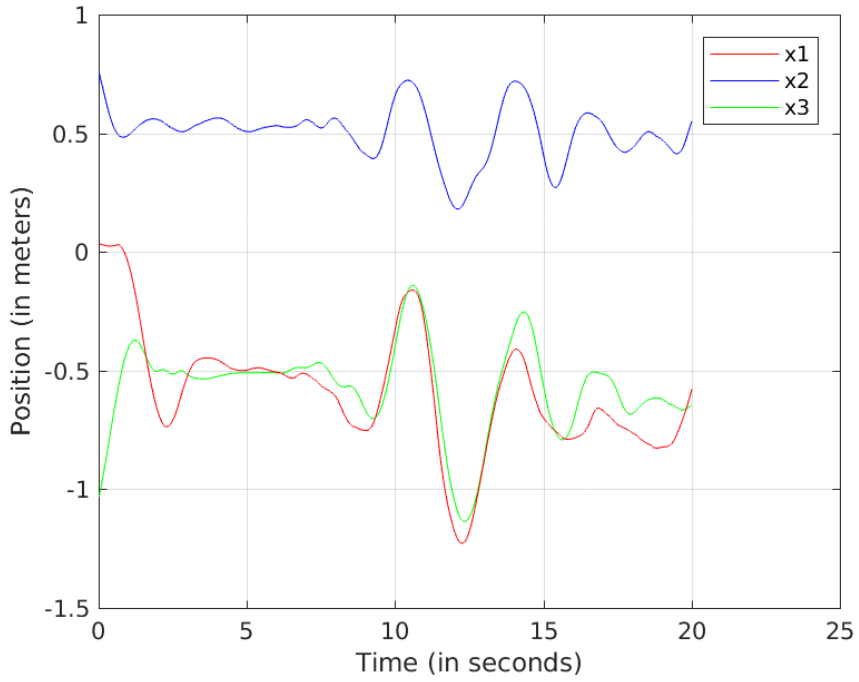

Fig. 4.4 Experiment 1, Position in x-direction.

[2] Z direction controller is not considered as it has different dynamics.

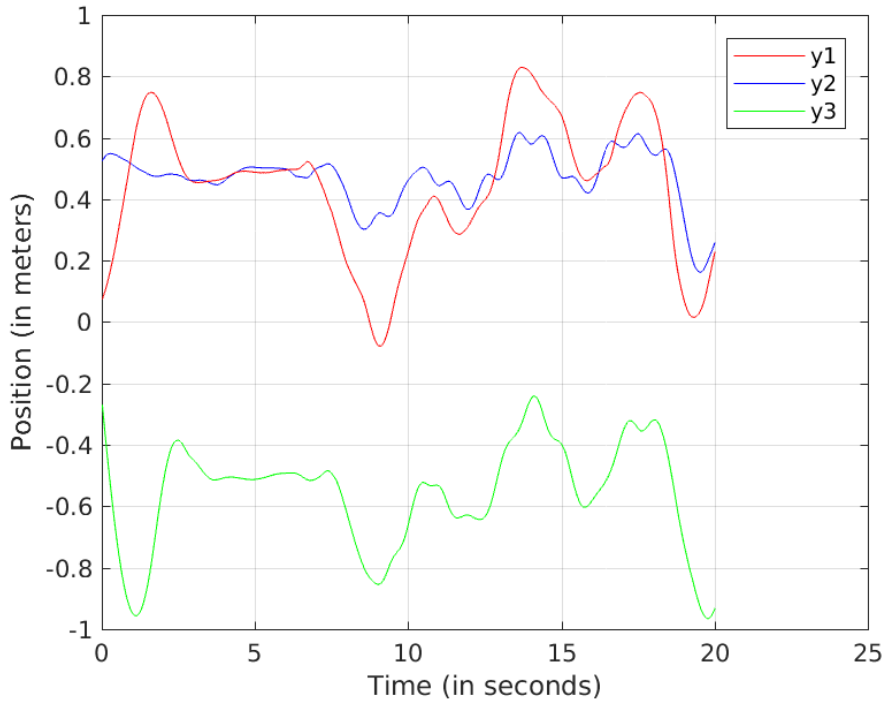

Fig. 4.5 Experiment 1, Position in y-direction.

4.2.2 Test flight 2

For the first test flight, the graph topology is

$$G = \begin{bmatrix} 0 & 3 & 2 \\ 3 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix}$$

Agent 1 is connected to the leader and other two agents follow agent 1 as per the graph topology.

The position variation of the quad-copters in X and Y directions are shown below

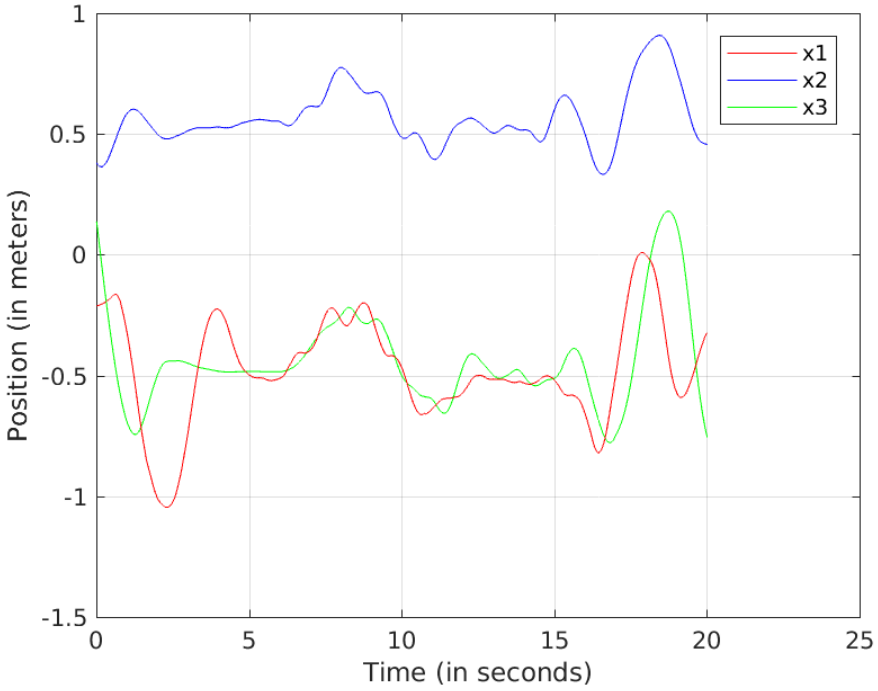

Fig. 4.6 Experiment 2, Position in x-direction.

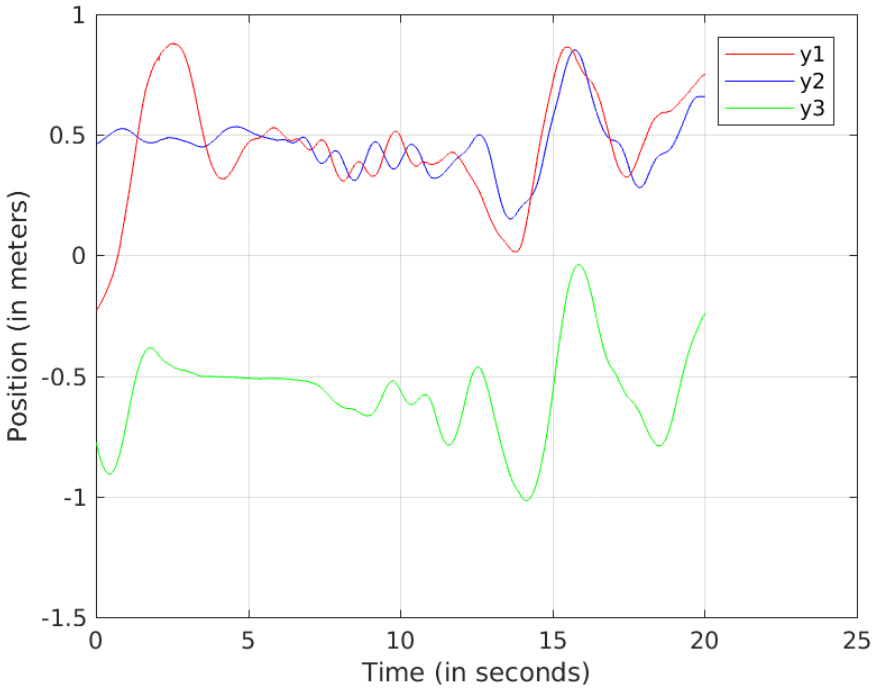

Fig. 4.7 Experiment 2, Position in y-direction.

4.2.3 Test flight 3

For the first test flight, the graph topology is

$$G = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

This time, all the agents are connected to the leader by using pinning gains

$$g1 = g2 = g3 = 1$$

The position variation of the quad-copters in X and Y directions are shown below

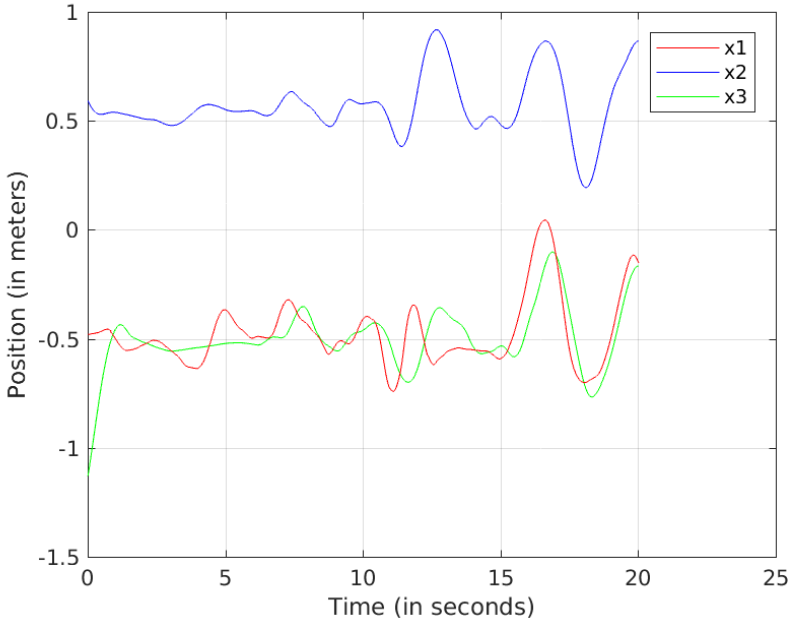

Fig. 4.8 Experiment 3, Position in x-direction.

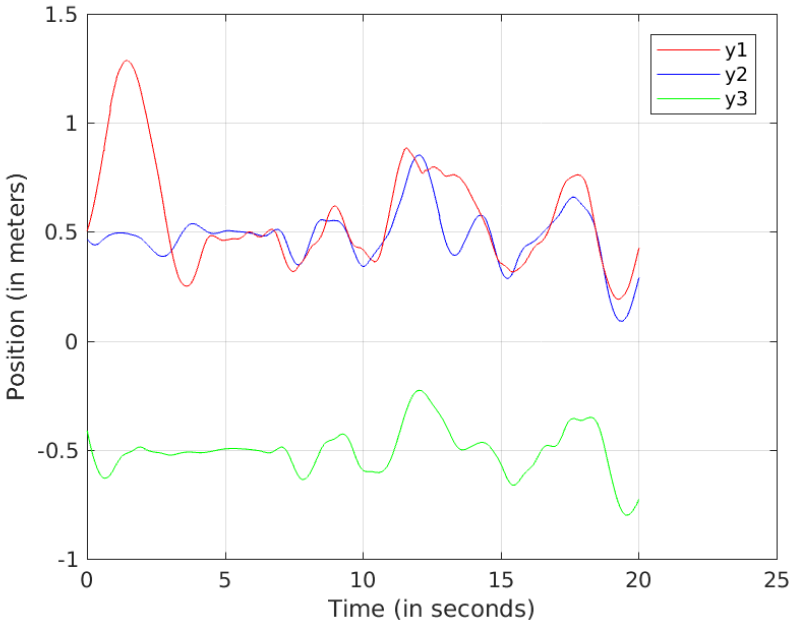

Fig. 4.9 Experiment 3, Position in y-direction.

Chapter 5 Conclusion and future work

From the test flights conducted, it is evident that with change in the graph topology the behavior of the swarm changes. With reduction in graph links, the latency in transfer of information increases in vice-versa. But the controller tries to minimize the positional errors anyways.

Also, it is concluded that the controller works for multi-agent systems and as it was tried on two systems with completely different dynamics, the controller also works for multi-class agents. Irrespective of what the initial topology of the network would look like, the controller tries to drive the agents such that the errors converge.

All the discussion above forms a small part of the big vision of establishing a mode of efficient and robust connectivity protocol between multi-class multi-agent systems. The simulations showed the convergence of the control algorithm for fixed and time-variant graphical topologies and the experiments verified the same. With this as the basis, future development should include developing dynamic leader switching and testing the time-variant topologies on some dynamical system of multi-agents with implementation of clustering where agents with different motives could choose to form separate teams and pursue individual goals. This could be done by appointing novel leaders within each new cluster.

Chapter 6 References

[1] Mashaghi, A.; et al. (2004). "Investigation of a protein complex network". European Physical Journal B. **41** (1): 113–121

[2] Grandjean, Martin (2016). "A social network analysis of Twitter: Mapping the digital humanities community". Cogent Arts & Humanities. 3 (1): 1171458

[3] Peter Sanders, (March 23, 2009). "Fast route planning". Google Tech talk

[4] Abraham, Ittai; Delling, Daniel; Goldberg, Andrew V.; Werneck, Renato F. research.microsoft.com/pubs/142356/HL-TR.pdf "A Hub-Based Labeling Algorithm for Shortest Paths on Road Networks". Symposium on Experimental Algorithms, pages 230–241, 2011

[5] Prim, R.C. (1957). "Shortest connection networks and some generalizations" (PDF). Bell System Technical Journal. 36: 1389–1401

[6] W. Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," IEEE Trans. Autom. Control, vol. 50, no. 5, pp. 655–661, May 2005.

[7] Joseph Hooper (June 2004). "From Darpa Grand Challenge 2004: DARPA's Debacle in the Desert". Popular Science.

[8] http://www.businessinsider.com/the-companies-most-likely-to-get-driverless-cars-on-the-road-first-2017-4/#1-ford-18

[9] V. Turri, B. Besselink, and K. H. Johansson, "Cooperative look-ahead control for fuel-efficient and safe heavy-duty vehicle platooning". IEEE Transactions on Control Systems Technology, 2017

[10] https://theloadstar.co.uk/psa-roadtest-autonomous-truck-platooning-singapore/

[11] https://www.bitcraze.io/crazyflie-2/

[12] https://www.vicon.com/motion-capture/engineering

[13] https://wiki.bitcraze.io/projects:crazyflie2:hardware:specification

[14] https://www.bitcraze.io/wp-content/uploads/2014/07/mcu_architecture1.png

[15] https://www.bitcraze.io/crazyradio-pa/

[16] https://www.vicon.com/file/vicon/tracker-3-11042017-55587.pdf

[17] https://www.vicon.com/products/software/tracker

[18] W. Lin, "Distributed UAV formation control using differential game approach". Aerospace Science and Technology, 35 (2014), pp. 54-62

[19] J Seo, Y Kim, S Kim, A Tsourdos, "Consensus based reconfigurable controller design for unmanned aerial vehicle formation flight" Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering. Vol 226, Issue 7, pp. 817 - 829

[20] P.Menon, G.Sweriduk, B.Sridhar, "Optimal strategies for free-flight air traffic conflict resolution". Journal of Guidance, Control and Dynamics. 22(2) (1999)202–211.

[21] Z.Qu,Cooperative "Control of Dynamical Systems: Applications to Autonomous Vehicles", 2008. IEEE transactions on automatic control, Vol. 53, No. 4

[22] J. N. Wang and M. Xin, "Integrated optimal formation control of multiple unmanned aerial vehicles", IEEE Transactions on Control Systems Technology, 21(5), 2013, pp. 1731-1744.

[23] L.S. Pontryagin, The Mathematical Theory of Optimal Processes, vol. 4, CRC Press, 1962.

[24] M. Simaan, J. Cruz Jr., "Sampled-data Nash controls in non-zero-sum differential games", Int. J. Control 17 (6) (1973) 1201–1209.

[25] G Xie, L Wang, "Consensus control for a class of networks of dynamic agents: switching topology", American control conference, 2006

[26] Jiang T, Baras J. "Graph algebraic interpretation of trust establishment in autonomic networks". Preprint Wiley Journal of Networks, 2009.

[27] T. Beth, M. Borcherding, and B. Klein, “Valuation of trust in open networks,” in Proceedings of 3rd European Symposium on Research in Computer Security – ESORICS’94, 1994, pp. 3–18.

[28] U. Maurer, “Modelling a public-key infrastructure,” in Proceedings of 1996 European Symposium on Research in Computer Security – ESORICS’96, 1996, pp. 325–350.

[29] T. Jiang and J. S. Baras, “Trust evaluation in anarchy: A case study on autonomous networks,” in Proceedings 2006 INFOCOM, Barcelona, Spain, April 2006.

[30] Y. Wan, S. Roy, A. Saberi, "Network design problems for controlling virus spread", *Proc. IEEE Conf. Decision Control*, pp. 3925-3932, 2007.

[31] Dhillon, Inderjit S., Guan, Yuqiang, Kulis, Brian, 2004. “Kernel *k*-means: Spectral clustering and normalized cuts”. In: Proc. 10th KDD, pp. 551–556.

[32] B. Kulis,S. Basu,I. Dhillon, and R. Mooney,“Semi-supervised graph clustering: A kernel approach,” ICML '05: Proceedings of the 22nd International Conference on Machine Learning, ACM, New York, NY, 2005, pp.457–464.

[33] https://www.neowin.net/news/fiat-chrysler-will-join-bmw-intel-and-mobileye-to-develop-autonomous-driving

[34] Lewis, F. L., Zhang, H., Hengster-Movric, K., and Das, A. (2013). “Cooperative control of multi-agent systems: optimal and adaptive design Approaches”. Springer Science Business Media.

Bibliographical information

Ankur Dalal, is a Mechanical engineer with research interests in applied robotics and controls. He has worked as a research intern at two labs at University of Texas at Arlington Research Institute (UTARI) where he worked on a humanoid robot (called PR2) and unmanned aerial systems. His work has mainly focused on development and application controls algorithm. Apart from this, he has also worked as robotics engineer at a drone delivery startup company, WAEC LLC., based out of Washington D.C., where his worked focused on development and implementation of navigation and mapping techniques for autonomous vehicles. He will be joining Aptiv PLC as a self-driving car engineer.