SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING
ELECTRIC POWER SYSTEMS GROUP

Self-Organising Smart Grid Architectures for Cyber-Security

A Thesis Presented for the Degree of Doctor of Philosophy

Calum Duncan Cameron

4/1/2017

Abstract

Current conventional power systems consist of large-scale centralised generation and unidirectional power flow from generation to demand. This vision for power system design is being challenged by the need to satisfy the energy trilemma, as the system is required to be sustainable, available and secure. Emerging technologies are restructuring the power system; the addition of distributed generation, energy storage and active participation of customers are changing the roles and requirements of the distribution network. Increased controllability and monitoring requirements combined with an increase in controllable technologies has played a pivotal role in the transition towards smart grids. The smart grid concept features a large amount of sensing and monitoring equipment sharing large volumes of information. This increased reliance on the ICT infrastructure, raises the importance of cyber-security due to the number of vulnerabilities which can be exploited by an adversary.

The aim of this research was to address the issue of cyber-security within a smart grid context through the application of self-organising communication architectures. The work examined the relevance and potential for self-organisation when performing voltage control in the presence of a denial of service attack event. The devised self-organising architecture used techniques adapted from a range of research domains including underwater sensor networks, wireless communications and smart-vehicle tracking applications. These components were redesigned for a smart grid application and supported by the development of a fuzzy based decision making engine. A multi-agent system was selected as the source platform for delivering the self-organising architecture

The application of self-organisation for cyber-security within a smart grid context is a novel research area and one which presents a wide range of potential benefits for a future power system. The results indicated that the developed self-organising architecture was able to avoid control deterioration during an attack event involving up to 24% of the customer population. Furthermore, the system also reduces the communication load on the agents involved in the architecture and demonstrated wider reaching benefits beyond performing voltage control.

Declaration

I hereby declare that this thesis is the a record of work undertaken by myself, that is has not been the subject of any previous application for a degree, and that all sources of information have been duly acknowledged

Copyright 2017, Calum Duncan Cameron

Acknowledgements

Throughout the course of this work several people have been instrumental in offering support, knowledge and wisdom.

Firstly I would like to thank my supervisory team of Professor Phil Taylor and Dr Haris Patsios for their tireless efforts in converting me, a graduate of computer forensics and software engineering into a power-systems researcher. Their faith in my abilities at points when I had almost completely lost it proved instrumental in completing this work. Their support combined with their dedication, expertise and knowledge of Viennese schnitzel restaurants have been instrumental in making a seemingly impossible dream possible.

Additionally I would like to thank the numerous colleagues at Newcastle University who have made this journey possible and ensured I got to the end more or less unscathed these people include: Ivan Castro-Leon, Barbara Alimisis, James King, Jialiang Yi and David Greenwood. Further thanks also to Charles Morisset and Zoya Pourmirza for the assistance in devising an appropriate cyber-attack strategy for the project. Further thanks to the Durham University CDT in Energy.

I would also like to express my thanks to my mother and siblings for surviving my company throughout this process which has not gone unnoticed, and for providing food, and a good internet connection.

Also major thanks to Rachel Miller, who despite going through a lot in the past year has remained unwavering, dedicated and very supportive even when everything seemed so bleak. This was even truer as I missed notable events and social gatherings in the pursuit of completing this work.

Finally I would like to recognise the cast and crew of both the Shoestring and Matchbox theatre companies, for accepting that the completion of this thesis and the work associated with it have meant that I haven't learned my lines as quick as I would normally do. Special mentions also go to Bill Wilkinson, Tom Casling and Ken Martin who have remained patient with me as I missed and arrived late to rehearsals to get more work done.

Publication List

Conference Papers

Calum D. Cameron, Phil Taylor, Charalampos Patsios, "Scalability in Smart Grid Architectures" 2014 49th International Universities Power Engineering Conference (UPEC), 2014, DOI: 10.1109/UPEC.2014.6934655

Calum D. Cameron; Charalampos Patsios; Phil Taylor, "On the benefits of using self-organising Multi-Agent architectures in network management", 2015 International Symposium on Smart Electric Distribution Systems and Technologies (EDST), 2015, DOI: 10.1109/SEDST.2015.7315231

Journal Paper

Calum D. Cameron, Phil Taylor, Charalampos Patsios, Zoya Pourmirza, "Using Self-Organising Architectures to Mitigate the Impacts of Denial-of-Service Attacks on Voltage Control Schemes", IEEE Transactions on Smart Grid (Submitted)

Contents

Figure List

Table List

Chapter 1: Introduction

1.1 OVERVIEW

The current power system has been reliant on centralised generation and fossil fuel resources, but an increasing demand for sustainable energy production is driving significant changes. This challenge forms part of the energy trilemma which outlines that the power system needs to achieve sustainability, availability and security. One of the proposed solutions to this challenge is the concept of a smart grid. This is often used as an umbrella term for a collection of methods, tools and technologies designed to operate the electrical network more intelligently and with a longer term view of electrifying additional sectors such as heat and transportation. As a result the power system is moving towards an energy landscape which is significantly supported by an increasing amount of ICT components. Smart grid research covers a wide spectrum of topics including data management, embedded intelligence and control algorithms.

The rise in electric vehicle usage and distributed generation through renewables has created a divergence from the traditional unidirectional power flow process, originating from large scale central generation and flowing to the distribution network. As a result the challenges faced by network management processes have become more complex, challenges which have encouraged the development of new techniques and technologies. Smart grids have also been cited as a method of deferring expensive network reinforcement programs through the use of intelligent systems designed to actively manage the current assets such that they are capable of handling the challenges of increasing customer demand and stochastic generation. Many of these intelligent systems are supported by new technologies in the form of energy storage systems, soft open points and demand side response approaches.

Plans for increased observability and controllability will result in the installation of a vast array of sensors, monitoring equipment and emerging controllable technologies. As a result these systems will produce and consume increasing quantities of data. This increase in data quantities produced by the number of devices present in the smart grid domain will need a suitable ICT infrastructure. In addition to the physical aspects of a communication infrastructure, the interaction between entries also requires appropriate design. This design considers the architecture involved with passing messages and control signals between controllable entities, sensors and controllers. When the communication architecture is hosting critical commands responsible for triggering protection devices or requesting frequency response the timing of the messages is crucial. An architecture which is unable

to manage data retrieved from sensors will likely be subject to delays in message transmission and therefore fail to deliver signals triggering protection measures in time.

However the inclusion of emerging technologies in the network management and data collection roles within the smart grid also creates more potential vulnerabilities. Cyber-security has played a pivotal role in a wide variety of industries adopting a more digitally oriented service, however industrial control systems have not followed the same path. In many cases this oversight has been the result of these systems being isolated from a wide reaching communication network and therefore access was limited. The increasingly ubiquitous nature of information and communication technologies within the power system and the movement towards an increasingly cyber-physical grid has created new opportunities for attackers.

The core message behind the developments in the smart grid domain is the requirement for flexibility. Customers are encouraged to be more flexible in terms of their consumption; control approaches require flexibility in terms of processing an increasingly dynamic set of variables. Therefore it is reasonable to expect that the supporting ICT infrastructure would also benefit from building flexibility into designs and implementations. This would take both the form of the communication infrastructure of routers, switches and wireless links, but also the population of sensors and controllers managing the physical network. Even in the presence of advanced forecasting techniques and predictive methods the amount of uncertainty within elements of the smart grid is estimated to increase and therefore it becomes more applicable to investigate and devise systems which can function under that uncertainty. Systems which have the capability to react and modify themselves in response unpredictable events such as a cyber-attack will become more relevant.

1.2 THE AUTONOMIC POWER SYSTEM

An advanced concept within the smart grid domain aims to encapsulate a range of research topics in creating an autonomic power system (APS); this concept as defined by the authors [1] refers to a long term vison of network automation. The APS is presented through the development of tools, techniques and technologies which aim to serve the needs of a future network, one which contains very different control demands and components in comparison to the networks presently in use. The following figure taken from [1] presented in Fig. 1.1 demonstrates the components and concepts defined within the APS project. The

figure illustrates that the system operates under the assumption of a decentralised control environment, formed by a series of connected control zones. One of the core properties of the APS is its ability to provide flexibility in the face of a highly changeable and uncertain environment and operates under a self* control paradigm.

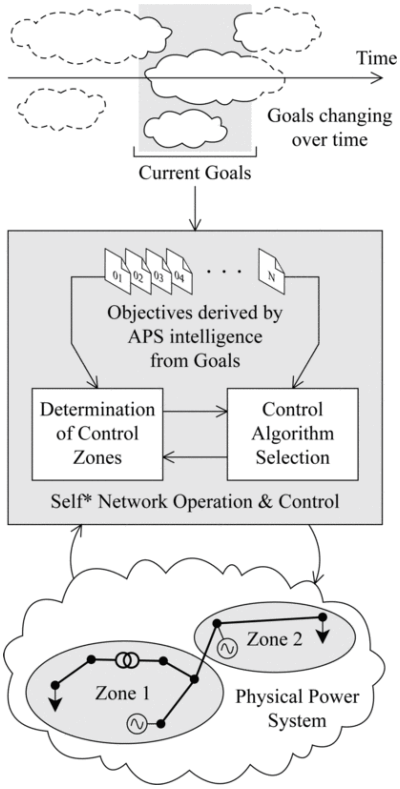


Fig. 1.1 – Autonomic Power System Outline Concept

The self* approach considers elements of the system to be self-optimising, self-healing, self-configuring and self-protecting, therefore being able to provide the desired level of flexibility a changing power system landscape will require. Within the zones of control themselves, a suite of control approaches can be deployed based on the requirements of the zone and the state of the network, demonstrating that flexibility and adaptability remain one of the core objectives throughout the whole concept. As the APS is a forward looking architecture it also considers greater controllability and observability than the present network with observability present deeper in the network. Consequently this will increase the amount of data and control signals which will need to be supported by the communication architecture, and ultimately increase the pressure on the ICT infrastructure. The timescale of the project would imply that the absolute performance of the communication channels would be capable of transmitting the volume of data and the processing power of the recipient controllers would not be overwhelmed. But, as previously

indicated a system with this degree of ICT integration would also include more potential weaknesses of points of entry for an adversary.

One of the potential mechanisms which can be applied within the control zones is that of multi-agent systems, which in turn can fulfil the objective of providing a decentralised control and communication architecture. The work presented in this thesis is in association with the APS project and therefore aims to deliver the flexibility and contribute to the overall self* process through the investigation into the potential for self-organising architectures.

1.3 MULTI-AGENT SYSTEMS

One of the goals of the APS project is to implement decentralised control approaches within individual zones, wherein each zone will be supplied with increased monitoring capabilities and more controllable components. This mechanism creates a need for a distributed problem solving approach which recognises each of the components in the system as individual entities which can contribute to completing control objectives. One method which allows the creation and management of interactive populations is multi-agent systems (MAS). The MAS approach involves communities of agents communicating and cooperating with one another being used to represent system components.

The concept of a multi-agent system is one where a collection of agents are in operation within a defined environment [2], and can be applied to a variety of differing problem domains depending on the classification of the entity representing the agent. Some MAS may consider a human user as an agent, others considering autonomous robots, alternatively an agent can be defined in software. A specific definition of software agents is contested within the research community, whereby individual cases have been defined based on the context of the scenario [3]. Individual authors define their software agents in terms of the problem they are designed to solve and therefore by the behaviours and goals installed within them. Agents can be physically located within sensors or embedded software in controllable devices, and therefore can be applied within the smart grid domain to achieve the desired level of observability and controllability.

In software based multi-agent systems the intelligence of the individual agents is based on their ability to complete tasks autonomously, negotiating internally within the agent community rather than resorting to relying on user intervention. The specific definition of

agent intelligence requirements depends on the problem domain, and the nature of the tasks and control objectives required. Some systems will require a degree of shared automation where a human user issues a command, but the agent population can calculate the optimum implementation of the command, this format may be applicable to fly-by-wire systems or robotics. Others will be supplied with a set of input parameters and conditions to satisfy and work towards achieving those goals without further input from a user. Although the applications and design approaches vary, a core set of characteristics is defined by the authors of [4]. These characteristics outline the requirements for a set of agents operating within the same overarching domain, and serve as a foundation for system development.

Autonomous – Demonstrating the ability to operate without human intervention and supervision, where the agents can come to conclusions and make decisions within the agent community.

Responsiveness – The ability of the agents to be able to respond to the changes within the environment, demonstrating that the agents exist in more than simply a monitoring capacity waiting for a user to affect the changes. In this instance the agents should be able to detect values exceeding thresholds and take action to resolve further problems.

Socially Active – This pertains to the necessity for communication between agents, each individual will not have all the information to make all decisions, and therefore will need to ask others in the domain. In addition to information gathering, agent communication is used to issue commands from one agent to another.

Proactive – A higher level of agent intelligence where, in addition to reacting to the conditions present in the environment, the MAS prevents threshold situations from developing, by making decisions ahead of a potential problem.

Other characteristics include an ability to learn from previous decisions, such that future decisions can be made with better knowledge [5]. In other cases some agents possess a degree of mobility, where they are able to move from platform to platform in support of standard communication [6]. This range of capabilities and applications indicates the flexibility of MAS and indicates that they can be applied to a range of scenarios; demonstrating that the core characteristics can be built upon when developing systems with increasing complexity.

One of the primary challenges facing MAS performance is related to scalability, according to [7] it can be considered to be one of the key limiting factors in the deployment of a multi-agent implementation. Within the context of power system monitoring and control – the interactions governing a single protection operation can consume 1.5GB of data an hour [8], which only accounts for one data source corresponding to a single control problem. If additional control problems are considered and further data sources are included, the data management issue within the MAS can escalate quickly. In the case of the APS vision, the objective of increasing observability and controllability will naturally contribute significantly to the volumes of data production and therefore have an impact on the agents involved. These impacts take the form of being able to process the information, make control decisions and disseminate the commands to other agents with the community. In addition to an increasing quantity of raw data, a growing agent population can produce scalability problems in terms of communication [9]. The net outcome is the increased risk of delays within data transfer between elements within the agent community.

1.4 CYBER-SECURITY ISSUES

The use of self-organisation provides flexibility to a system operating in a dynamic control environment, where the integration of distributed generation, electric vehicles and energy storage present emerging monitoring and control challenges. But in addition to the challenges posed by the properties of the APS network, the technologies involved create vulnerabilities which can be exploited by cyber-attacks and therefore creates a reliance on cyber-security.

Cyber security refers to a set of tools, techniques and procedures which are employed for the purposes of protecting any element or elements of a computerised system from damage or intrusion. These systems have become a fundamental part of the digital era, and products and services relating to protecting data or personal devices have become almost as ubiquitous as the technologies they are designed to protect. However as stated by the authors of [10] any network security mechanism exists under threat from a potential attack event where an adversary or adversaries intends to damage the network, extract confidential data or manipulate information transmitted between components.

As modern power systems grow and evolve towards the target of becoming smart grids, the additional monitoring, control and communication technologies required to achieve that

goal also increases the number of potential points of vulnerability [11]. The threat of cyber-attacks is not limited to the realm of future network solutions and technologies, design choices made during procurement and development of current SCADA systems installed during the mid-1990s and early 2000s lead to a concept of openness where few considerations were made for system security [12]. One of the reasons that new and emerging smart grid concepts may become increasingly vulnerable is the transition from a small number of controlled devices to a widespread interconnected network [13]. Furthermore the authors of [11] indicate that several smart grid access points are low cost devices with limited security provisions and are at risk from malicious tampering.

Several different cyber-threats have been investigated with respect to the smart grid research domain, and several different targets have been identified. For example several papers have been published on the topic of false data attacks - [14] [15] [16], this attack format involves manipulating messages produced by monitoring devices to misrepresent measurements. Therefore control processes begin performing state estimation and control decisions based on an incorrect representation of the systems' properties. Other research conducted by the authors of [13] considers a denial of service attack whereby elements of the smart grid network become inundated with a stream of noise messages. While researchers have suggested the creation of techniques to detect the presence of false data, the authors of [13] explain that other than purchasing more bandwidth there are limited solutions to the problem of denial-of-service attacks. The authors of [17] consider the aspect of cyber security from the perspective of preventing unauthorised access to substations which can host networked devices and servers. Most substations are unmanned and have limited physical security mechanisms and therefore present further vulnerability whereby an attacker can gain physical access to control hardware and software. A further attack approach can be implemented through the use of malware,

In addition to the smart grid being vulnerable to a range of different attack methodologies it also plays host to several potential targets which may be of interest to an adversary. For example threats against SCADA systems are considered by the authors of [18] and [19]. As the SCADA system contains data reflecting a wide area of the network, an attack has potential for influencing or triggering blackouts akin to those described by the authors of [20]. Other functions at risk from a cyber threat include state estimation functions [21]; state estimation is responsible for the provision estimating voltage magnitudes and angles at key buses. However it is also used in power markets, forecasting and contingency

analysis among other functions, an attacker can prevent the state estimator from building an accurate model of the network through the injection of false data. As a result incorrect control signals can be issued as a result of the estimator indicating the presence of faults or anomalies in the system which bypassed bad-data filtering processes. A method for influencing the state estimation involves compromising Phasor Measurement Units (PMUs) such that they begin to transmit false measurements as described by the authors of [22] and is also considered from the perspective of a network of sensors involved with the transmission of false data by the authors of [23]. The attack mechanisms discussed have also been suggested with the impact of damaging voltage control procedures

In addition to focussing on the attack strategies research is also drawn to the detection and analysis of cyber threats. For example the authors of [24] present a method for detecting attacks against voltage control processes though the analysis of the sources of bad data. The source of the attack against voltage control is centred on the falsification of measurements and therefore triggering unnecessary tap changes which can shorten the lifespan of the transformer. A similar consideration is applied when taking into account overall system reliability [19] which surrounds applying defences and security protocols at the SCADA level.

Further work considers threat analysis as presented by the authors of: [25] [26] [27]. The authors of [25] explain that the supporting ICT infrastructures involved with operating a smart grid solution do not go through the same rigorous simulation and analysis that the electrical elements do. Therefore understanding impacts of outages and attack induced failures is weaker than it is in comparison electrical outages. The consequence of this is that vulnerabilities may not be detected and the impact of those vulnerabilities is not fully considered and as a result the correct actions required to close the vulnerability are not taken. Furthermore the authors of [26] indicate that the processes involved in modelling an attack event from the perspective of its impact on the physical system is limited. These limitations are a result of modelling techniques having limited hardware integration and end-to-end system modelling. As a result the ability to ultimately assess how an attack event would influence a given network is restricted. The authors indicate that simulation is restricted by the degree of variance involved in the cyber-security domain

The impacts of such an attack on the power system can have significant consequences for performing control and potentially lead to physical component damage, financial losses

and outages all triggered by loss of control signals. This underlines the significance of being able to devise solutions and methods for mitigating and limiting the effects of this format of attack strategy. Cyber-attacks have been specifically targeting power systems, for example Stuxnet [28], Israel [29] and Ukrainian [30] events. The latter containing a denial of service component to the attack strategy. Therefore cyber-security issues are a prominent driver for developing control and communication architectures which can offer robustness, as the authors of [25] documented electrical networks are heavily analysed, but the supporting architectures are not.

1.5 SELF-ORGANISATION

The challenge created by the threat of cyber-attacks is one that is the responsibility of the supporting ICT infrastructure. That infrastructure may contain a multi-agent system operating under the jurisdiction of the APS and therefore tools and techniques involved in forming a defence against a cyber threat will need to be integrated such a system. One approach is the use of self-organising systems. A self-organising system can be defined as one that can satisfy the requirements of: scalability, robustness, flexibility and adaptability [31] and one that in distributed systems where the set of components and their interactions change and evolve in response to the problem domain [32]. The concept of self-organising architectures is not a new idea within the field of computer science, with specific reference to computer networks and P2P configurations as noted in [31]. A concept for self-organising software architectures was initially proposed in 1996 [32], [33]. As this is a research topic that originated within computer science with respect to network interactions such as [34], a volume of research considers self-organisation from a network perspective. However these concepts are being increasingly applied to wider subject areas – solving business energy consumption levels [35], and transportation networks featuring intelligent vehicles [36]. In alternative applications the process of self-organisation is focused on modifying the internal behaviours and how they approach their goals and objectives as documented by the authors of [35]. According to [37], the concept of a self-organising architecture or network has several key properties.

The first property is scalability, whereby both natural and engineered systems achieve scalability through two main requirements. Firstly a lack of complexity – using a set of simple behaviours aims to prevent the number computational demand increasing as the number of system components increases. The second driver in achieving scalability is a

focus on local decision making instead of a centralised approach. Using local, simple rules prevents a significant growth in operations for an increasing population as local controllers are only exposed to local population changes not global increases.

The second property centres on stability – wherein the network or agent community must remain stable when transitioning from one configuration to another. The stability property also encompasses the idea of robustness to the extent that in the process of making a transition from one configuration to another the functionality is not lost. The third property is agility, which describes the ability of the system to transition from one configuration to another within a reasonable timeframe. Furthermore the transition must not be an over-reaction which would result in an oscillation in network states.

These principles are taken from a paper discussing cellular communication networks – but the general property requirements can be applied to self-organising architectures in smart grid control and monitoring. The three properties of Scalability, Stability and Agility are all relevant performance factors when transitioning between configurations. A system with strong scalability will likely offer strong agility as slower transitions could be influenced by issues pertaining to the provision of scalability. Furthermore the less time the architecture spends in an intermediary inter-transitional state reduces the likelihood that communications will be transmitted before all the connections have been formed resulting in greater stability. Additional research completed by the authors of [38] introduced further properties which include the ability to form initial structures and connections automatically, and the ability to perform self-monitoring.

The Authors of [37] also compares various classifications of self-organisation, classifications which define the level of adherence to the self-organisation concept.

Adaptive Networks – Configuration changes are made in direct response to changes in system state, no consideration for scalability or agility. Triggered based on fixed threshold values

Autonomous Networks – Similar concept to adaptive networks, but implemented autonomously – without human or external intervention. One of the core components of a self-organised system but on its own doesn't fulfil the three primary principles.

Cognitive Networks – Autonomous networks with learning capabilities, such that trigger points are learned based on interaction with the environment. Cognitive networks maintain a level of interaction across layers in the system to facilitate the learning process.

Self-Organised Networks – Take traits from adaptive and autonomous networks and add continuous monitoring to decide the appropriate network transition, with the potential to lean from the decision for future reference. An advanced self-organised system would also be capable of self-optimisation and self-healing.

The work presented in this thesis includes the development of a self-organising architecture which takes advantage of continuous monitoring techniques, a self-initialisation stage and a decision making engine. Therefore the final solution can be classified as a self-organising architecture.

1.6 RESEARCH OBJECTIVES

The objective of the research presented in this thesis is to investigate the potential for self-organising architectures within a cyber-physical power systems domain and their role in providing robustness through cyber-security. At the conclusion of this investigation a self-organising architecture was developed to address security concerns in the smart grid domain and deliver resilience in the presence of a cyber threat which could not be provided by a static architecture design. The core objectives in achieving this target are outlined below.

1. Evaluate comparative performances across differing control and communication architectures in the context of distribution network management with a view to determining the potential role for implementing self-organisation. This investigation aimed to determine what the benefits would be of providing self-organisation within the control and communication architecture and why self-organisation is an appropriate approach for cyber-security.

2. Develop and implement an agent population with functioning self-organisational properties including architecture formation, contemporaneous monitoring and decision making.

3. Examine the performance of the developed self-organised system in the presence of external network threats in the form of cyber-attack events with respect to control

and communication performance. These performances are also examined with respect to a static architecture undergoing the same cyber-attack conditions with the objective of learning which variables are affected by an ongoing attack. A further learning outcome is to identify whether a communication variables have an impact on the electrical performance of a network while under attack. To determine whether the self-organising architecture can improve electrical performance by improving communication layer performance.

1.7 THESIS OUTLINE

The remaining chapters of this document will be as follows:

Chapter 2 will examine the concept of the power system as a cyber-physical system, investigating the role of ICT in the modern and future system operation. Further to this the chapter will continue to discuss the role of multi-agent systems as one of the techniques considered for deployment in smart grids heavily influenced by the integration of ICT solutions. The section will also cover examples of control and communication architectures implemented smart grid environments and the trends that can be extracted when looking at the development of such a system. Finally the section will then discuss the potential vulnerabilities that a power system with a high dependence on ICT can face, with a focus on cyber-security issues.

Chapter 3 presents the work completed as a series of control and communication architecture designs were investigated in a static state. The goal of the investigation was to determine the performance advantages and disadvantages associated with the architecture designs and explore the potential for implementing a self-organising architecture. The chapter documents the results of those experiments in terms of responding to a voltage deviation across increasing agent populations and under the influence of a cyber threat.

Chapter 4 presents a review of literature, investigated with respect to the design and usage of existing self-organising system applications across a range of research domains. The purpose of the review was to examine a series of techniques which could be adapted and for the implementing within the smart grid domain.

Chapter 5 presents the development of the implemented self-organising architecture. The chapter documents the three stages of operation and the components involved in the developed self-organising architecture. The architecture transition processes are also described along the method with which the cyber-attack events were triggered.

Chapter 6 documents the development of the decision making engine which was responsible for the interpreting performance monitoring data and converting that information into a transition action. The chapter indicates the initial creation of a decision tree approach which was then replaced by a fuzzy based decision making engine which was more appropriate when processing ambiguous input data in an environment with multiple sources of uncertainty.

Chapter 7 illustrates the evaluation framework used to demonstrate the effectiveness of the self-organised approach, including the design of the underlying electrical network, agent settings and attack conditions. The chapter also includes introduction of the integrated test environment used for connecting the MAS in JADE with a power system load flow engine in Matpower.

Chapter 8 contains the results from the sequence of test performed on the self-organising architecture where electrical and communication performances were compared with respect to a static architecture.

Chapter 9 is a discussion chapter where in the tool, techniques and approaches taken throughout the path of conducting the research are analysed. The chapter considers the wider applications of the research contributions and makes suggestions for future work within the research area.

Chapter 10 presents overall conclusions from the completed research, documenting the key findings, and evaluating the outcomes in comparison to the initial research objectives.

Chapter 2: The Cyber-Physical Power System

2.1 INTRODUCTION

As reliance on renewable energy sources and distributed generation increases, new challenges arise in monitoring and managing the network. Observability is notoriously limited at the lower voltage levels [39] consequently this leaves gaps in the ICT infrastructure governing monitoring and control of distribution level assets [40]. Due to the emergence of smart grid solutions featuring a cyber-physical infrastructure the volume of data present within the network is set to increase. Incorporating emerging smart technologies, energy storage, and electric vehicles produces an interesting problem for data collection and management.

As the role of ICT within power system control becomes increasingly important, especially with the growing interest in smart grid technologies, new solutions are needed. This promotes ICT concepts such as cyber-security and cloud computing further into the traditional power system domain. Approaches including Multi-Agent Systems (MAS) and self-organising architectures are a potential solution to aiding a developing energy network as it becomes more distributed and decentralised. Control algorithms within smart and micro grid contexts have been evaluated in terms of control speed and accuracy [41] and economic gain [42]. Additionally the ICT challenges facing future power systems are known [43]. However the authors in [43] and in [44] suggest that as potential MAS implementation offers suitable scalability for a microgrid control scenario.

The control architecture responsible for hosting a potential agent based solution and the volume of data associated with it will be subject to several design criteria. For example a multi-agent architecture can be heavily influenced by scalability as suggested by the authors of [7]. In MAS design scalability is considered to be a product of two factors – complexity and load [45]. The problem of complexity is the more dominant driving force in existing research rather than data processing capacity and delays. Scalability assessments have been made of MAS platforms outside of the power systems domain [46], [47] indicating that MAS are capable of handling large volumes of data. However these investigations were conducted within the context of a fixed number of messages, instead of supporting continuous data input in a real-time control environment.

It is envisaged that future grids will be more reliant on decentralized control architectures, mainly due to increased penetration of distributed generation and the customer's active role in the energy market. Decentralised cells or zones of control as discussed in [48] and [49]

are used to account for the growth in complexity and in uncertainty of power system control problems. A decentralised network featuring smart technologies and distributed generation presents new control requirements and challenges. Multi-Agent Systems (MAS) are often suggested as a potential distributed control solution for the projected decentralised power network. These systems are not only governed by the communication infrastructure linking network nodes to one another but the configuration of the communication hierarchy. MAS have been suggested for use in power systems control for various solutions - for example [50] and [51] examine multi-agent solutions for microgrid control – where agents equate to components within the microgrid.

2.2 MAS FOR POWER SYSTEMS

The Power System industry has adopted MAS for the purposes of research in emerging smart grid approaches; the predominant driver for this research in these investigations is the application of multi-agent systems for control. An overview paper illustrates the relevance of MAS within Power Systems and a series of requirements for a fully operational multi-agent control approach Part 1 [52] and 2 [53]. These papers document the importance of MAS within the power system domain and that research interest has been growing since 2001. The primary focus of that research lies in control procedures, using agent based solutions to derive solutions to control issues within the power system domain – or in the simulation of complex scenarios. Subsequent research has been drawn to specific case studies or individual control challenges. For example the authors of [54] examine the potential for MAS in microgrid control; the paper introduces a control approach implemented through MAS where each agent in the system equates to a component within the microgrid. While the solution demonstrates the applicability of MAS to power system problems, the issue of scale isn't actually considered beyond the abstract of the paper. This suggests that there is an interest in making MAS based control solutions scalable, but specific contextual research is limited. It has been determined that MAS have the potential to operate on a larger scale, but the consequence on performance of a real-time system lacks coverage. Another example is [9] where the impact of scale is mentioned through the problem of communication demand yet the number of agents at which this problem is encountered is not mentioned; nor is the impact on controllability as the system approaches this population. In this instance the communicative approach is different to some of the other methods in the domain, as messages are sent to a server from the points of

measurement instead of via the FIPA-ACL protocol. Some of the scalability problems may have been associated to this approach, as the ACL had already been analysed for its efficiency within the same agent platform (JADE) [55]. However interactions between agents in a power system environment across different hardware will need some communication – likely internet based – to disseminate information and commands. Note that both the previous examples are aimed at the power system in an islanded state which is a representation of MAS as a closed environment to which interactions are encapsulated within the specified control domain. This example suggests that the MAS assumes control in the event that the target network becomes islanded as part of the restoration process [56]. Here MAS devises an island microgrid after a fault event has occurred, it is a solution which relies on the presence of storage within the islanded section of the network in order to meet the demand requirements of those customers in the island. It is not clear whether the MAS remain an intrinsic part of maintaining control in the case study while a fault condition has not arisen or after the fault has been cleared and the island reintegrated into the grid.

2.3 MAS PLATFORMS

In a survey of "Multi-Agent Systems" within IEEE journal publications (Smart Grid, Power Systems and Sustainable Energy) of those in which an agent based implementation was featured – the primary development tool was JADE (7 out of 16 papers 2010-2014) as presented in Table 2.1. The nearest alternative featured mathematical abstractions of an agent population (3 MATLAB models and 3 algorithmic representations) for the demonstrative purpose of control algorithms, agent discovery and research into power system markets. These alternatives represented the agent community as a series of networked nodes through graph theory, and do not necessarily represent an agent community which operates in accordance with the FIPA standards.

Table 2.1 – Survey of MAS Platforms

	Agent approach					
year	JADE	Mathematical Model	Matlab Model	EPOCHS	KQML Agents	JACK
2014	1	0	3	0	0	0
2013	5	0	0	1	0	0
2012	0	1	0	0	1	1
2011	0	2	0	0	0	0
2010	1	0	0	0	0	0
Totals	7	3	3	1	1	1

The two divisions of analysis are often governed by the difficulty in unifying MAS development software with power system simulation engines – each one requiring an abstracted form of the other to generate results. Several instances have connected JADE

with alternate software packages such as MATLAB/Simulink - [57] [58] [59] InterPSS [60], and PSCAD to introduce deeper modelling into the process. However the larger problem lies in the differentiation in the two opposing forces, a JADE development is effectively a multi-threaded software package which obeys and adheres to the communication and operational principles of the accepted FIPA standards. Therefore it runs in real-time, allowing data flow and communicative response times to be effectively measured. However an associated power-flow/modelling engine operates on the principles of time-steps and incremental assessments of the system state, achieved by passing sets of variables between the MAS and the power system model. This can test the ability of the agent community to solve complex problems in a co-ordinated manner when provided with valid information, but often omits the communication load and data management issues present within the communication architecture.

Some implementations where agent co-ordination is the primary goal utilise JADE as a stand-alone solution without developing an interface with a second application for further modelling [61]. Yet contrasting work considers MAS involvement from a theoretical perspective, using the agent principles of inter-connectivity as base material [62].But the general consensus is that for the agent development phase of a research activity, JADE appears to be the most prevalent platform of choice within power-system research topics.

2.4 CONTROL AND COMMUNICATION ARCHITECTURES

Literature in MAS architectures as presented by the authors of [63] illustrates the intrinsic connection between the context of a MAS implementation and the control/communication structure. Certain designs such as the hierarchy and the federated architecture illustrated in Fig. 2. are more applicable to a problem domain with an inherent embedded structure such as a smart microgrid.

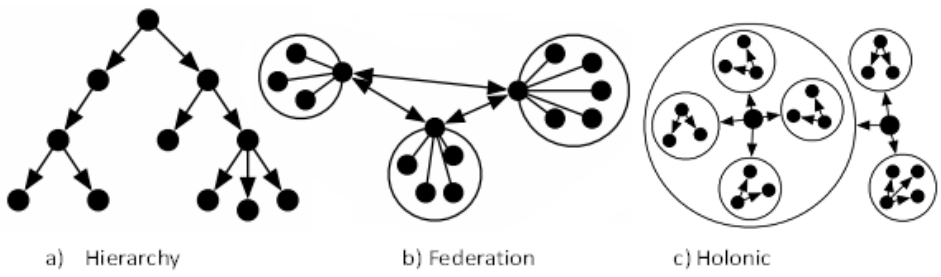


Fig. 2.1 – Multi-Agent Architectures

A hierarchical structure as illustrated in Fig. 2.a, can support a variety of control approaches but as the literature has indicated is prone to hosting several tiers of control at once to cover different aspects of the problem. The overall hierarchical concept is one that does conform to the conventional structure of the electrical network, whereby smaller components are positioned at the bottom of the architecture, transformers occupy the intermediary layers and the primary substation is the head of the structure. If the architecture was governing a higher voltage level the steps within the hierarchy would change and contain differing components. Communication in this environment does not pass between nodes on the same hierarchical tier and is only transmitted vertically to other tiers.

On the other hand the federation structure in Fig. 2.b is aimed more towards local sub-domain controllers – where the head agent serves as an intermediary between a cluster of agents and the rest of the community. In this format the head agent contains greater responsibilities than many of the nodes in the hierarchical format; this represents a more complex structure whereby individual head agents can communicate with each other unlike in the hierarchical structure where communication was only vertical. As a result local controllers will be able to work with one another to solve network problems and may be more appropriate when dealing with global network challenges rather than local incidents such as performing frequency control. A final example considers the holonic architecture presented in Fig. 2.c, the holonic architecture consists of a series of nested sub-domains which exist as independent cells of the overall architecture with their own goals and intentions but also fit together to operate a larger system. Within power systems holonic multi-agent systems have been suggested to offer a number of benefits in smart grid operations, including autonomy, dynamic reconfiguration and security considerations as indicated by the authors of [64].

In addition to the core architecture information from a MAS perspective it is also important to consider the architecture implementations from a power systems perspective, so that common traits can be identified. Furthermore the investigation intends to discover to extent to which self-organisation is considered in existing implementations. With the growing trend and interest in decentralisation and distributed energy multiple examples are present within the power system domain, especially as emerging research focuses on developing and investigating the grid as a cyber-physical system. The first example stems from a paper entitled: “Scalable Distributed Communication Architectures to Support Advanced Metering Infrastructure in Smart Grid” [65]. The paper describes potential communication

architecture within the context of the smart grid, where smart-meters are considered to be the lowest hierarchical entity and therefore make up the bulk of the population. The authors examine the issue of architecture design in two phases in the event of the deployment of an advanced metering infrastructure – which drives the necessity for communication architecture design.

The initial diagrams taken from [65] and presented in Fig. 2.2 illustrate communication architectures associated with data collection from a smart-metered customer population. In the diagrams Concentrators receive several input signals from the customer population and relay a single signal to the operation centre. While the paper is aimed at communication architectures, the initial diagram in Fig. 2.2a, implies a centralised control methodology oriented around a central database structure. The authors focus the attention of the paper on the processes involved with data management and collection rather than the implementation of control algorithms, in the initial structure all data processing is conducted centrally

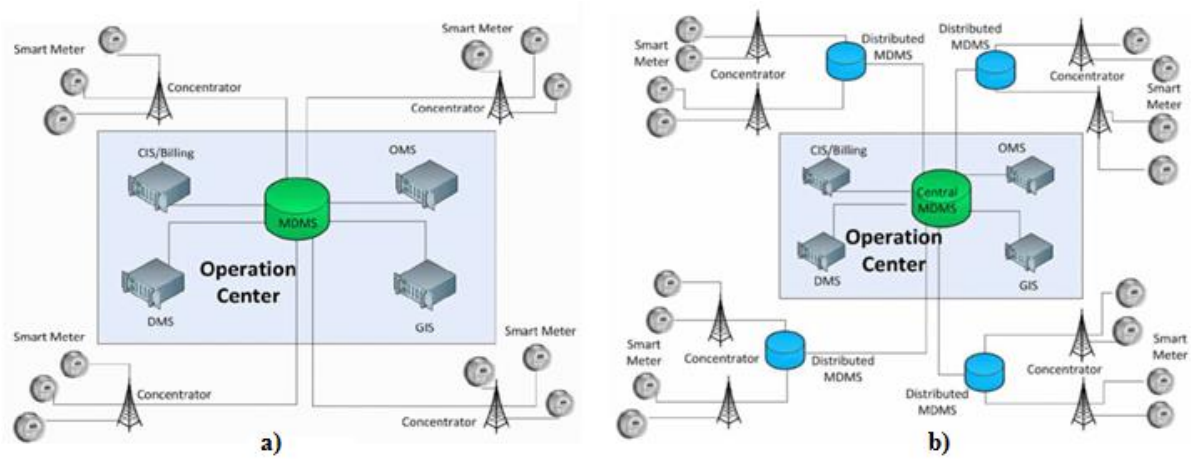


Fig. 2.2 – Traditional and Alternative Advanced Metering Infrastructures

The paper goes on to suggest improvements using the objective of scalability performance. In Fig. 2.2**Error! Reference source not found.**b, additional Concentrators are suggested per quadrant. In effect this becomes a closer partner to the Tiered or Clustered Architectures where the Distributed MDMS placements act as upper-tier aggregates. The addition of more Concentrators acts in the same way as the clustering process used in both the Tiered and Clustered communication architectures. The purpose is to relieve the load pressure on the aggregation points by dividing up the set of connected agents or entities. The central operation centre remains a focal point for any decision making processes, the Distributed MDMS entities conduct local level information processing but the information is ultimately

still passed to the operation centre to be interpreted by the various management systems. Because communication load is reduced at the concentrator and at the D-MDMS, the bandwidth requirements and costs are reduced.

A third approach is suggested, where a second tier of data collection points contained within a series of Distributed Operation Centres as presented in Fig. 2.3 .The Distributed Operation Centres process localised control decision making in addition to data collection. The central operation centre retains some control responsibilities and the paper references the consistency of the interface between the Operation centre and the distributed operation centres. As scale increases, through the inclusion of additional smart-metered customers, the operation centre retains the same number of interactions, as the Concentrators and the Distributed Operation Centres absorb the additional load.

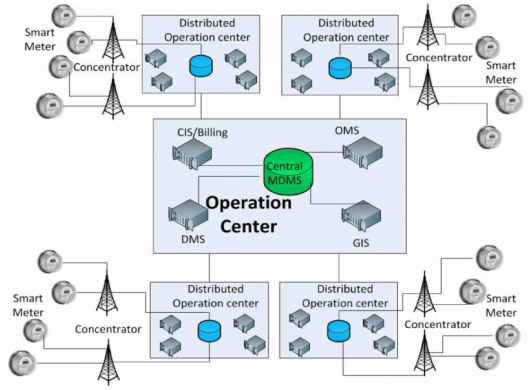


Fig. 2.3 – Advanced metering infrastructure with additional local controllers

However although three solutions are presented within the paper the authors do not indicate a mechanism which would consider switching between the different configurations. Each of the architectures illustrated is intended as a static architecture without the abilities and knowledge required to perform a transition event.

A second paper [66], examines the collection of data from PMUs across a wider geographical area than the first example, using substations rather than smart-meters as the smallest entity. It indicates that due to the various latency requirements of different system events, the placement of control within the system architecture would depend on the nature of the control problem. For example applications for responding to transient stability issues would need to be placed close to the source as the response times need to be within 100ms. Alternatively applications focussed on voltage stability or post-event system analysis can be positioned more centrally. Fig. 2.4 illustrates the two contrasting approaches for data collection and dissemination of control signals. The first case represents a centralised

structure, wherein all substations communicate with a single controller overseeing the entire monitored population. The authors indicate that such a configuration could be useful in terms of injecting control commands directly into the substation without navigating through a series of intermediaries. The connection between the two is not anticipated to be a direct link given the potential geographical distance between source and controller but routed through a communications network involving several routers but those communication hops perform no processing. The authors indicate that the centralised approach may be more suitable to less time-dependant applications such as voltage stability and state estimation – wide area management tasks. But the architecture may prove unsuitable for fast response applications such as protection wherein responses are required in the time frame of a few cycles. The issue surrounding the response times is due to scalability and the number of connections with the central control centre producing a greater workload and ultimately influencing response times.

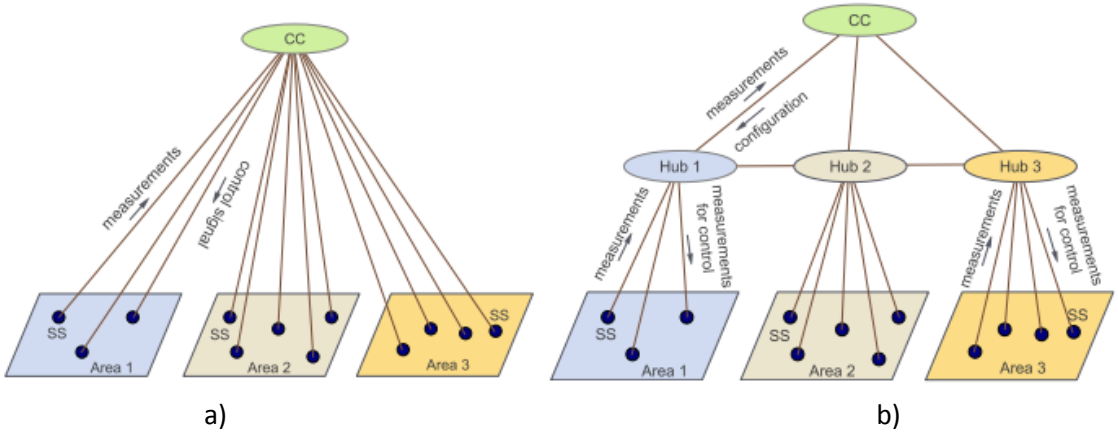


Fig. 2.4 – Centralised and Decentralised Communication Architectures

Fig. 2.4b illustrates an alternative approach which features intermediary Hubs which retrieve data per network division. As documented in the paper, these hubs do not contain any controlling functionality – local control algorithms are in place at the substation level and wider area management tasks are completed by the central controller. The purpose of the Hubs is primarily data management and routing, such that the control centre is not overloaded with incoming communications. Hub connections are physically located agents and are affiliated with specific buses within the electrical network – indicating that non-component agents remain part of the physical architecture. As the network sections at the hub layer are electrically connected, there is a case for communication between agents in the same tier and stepping away from a hierarchical system with a purely vertical communication structure. The authors also imply that the control centre can change the

configuration of this architecture depending on the power system conditions – while the concept of self-organisation is briefly hinted at, supplementary details on how the architecture would change and what the alternative configurations would be are not documented in the paper. Each of the network divisions then communicate with the central controller. Table 2.2 provides an insight into the scale and composition of the architectures involved – not all of which constitute a communicative entity within the architectures. It indicates that the network divisions are roughly even with Area-2 being the largest network division.

Table 2.2 – Network Division per Architecture Design

Parameters	Topology 1: Centralized	Topology 2: Distributed		
		Area-1	Area-2	Area-3
Buses	118	37	45	36
Substations (SS)	109	34	40	35
Control Centers (CC)	1	0	1	0
Data Routing Hubs	0	1	1	1
Generator SS (GSS)	52	16	19	17
Communication Links	161	50	59	52

This indicates that in the case of the second topology the distribution of connections intends to remain fixed; in the eventuality that one of the hubs becomes congested there is no immediate mechanism for transferring substations from one hub to another. As a result the hinted self-organising concepts will only apply to control functionality and agent responsibility rather than communication architecture changes. Furthermore there is no redundancy in the system in the case of hub failure when using the second topology. This is more severe in the case of the first topology as there is a single point of failure, therefore in the event of a cyber-attack on the central controller – all of the connected substations and by extension a large number of customers will be affected.

A third example of architecture design within the power system and smart grid research domain aims at implementing autonomic computing principles as part of the system design [67]. The structure presented in Fig. 2.5 encompasses 2316 LV customers (red rectangles on the diagram) spread across several intermediaries. The customer population is divided between two towns connected at the exchange point below the grid controller – the leftmost branch includes 1679 customers and the right branch the remaining 637. The customers are grouped into local clusters of 58 customers each under the supervision of a Transformation Connector (TC), and the TCs in turn are then grouped into subsets of 5 TCs per one Demand

Manager (DM). The TC forwards information from the customer to the DM, and the DM enacts balancing policies and responds with control signals to be passed via the TC to the customer. Additional architectural structures in the form of sector connectors separate different groups of customers from one another, so that policies can be applied to a specific classification of customer. This configuration is considerably more deconstructed than the other examples investigated thus far. Like the first example, the ALR-SCM network considers the customer to be the minimum entity for monitoring – while the network presented in Fig. 2.4 considers substations as the smallest entity.

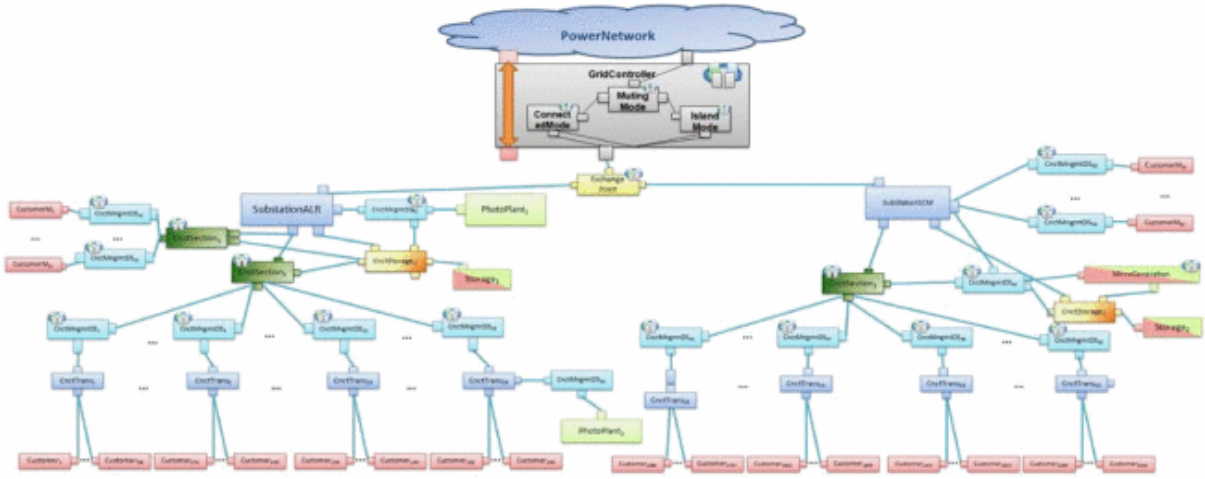


Fig. 2.5 – Autonomic System Architecture

In practice, microgrid control scenarios would require multiple layers of control, because an active network has multiple variables to manage simultaneously – as noted in [66], the location of the control can be dependent on the timescale the control operates within. Control objectives with a longer time horizon can be processed by an agent further away from the agent under control, whereas more critical objectives requiring faster responses would be hosted close to the controllable entity. The purpose is to limit the influence of latency and interference on critical control signals.

In that respect the set of requirements desired by a particular network and control condition may differ from one another. For example a power restoration problem as presented in [68], is centred on the agents at the lowest part of the network being the first responders. Post-fault the customer agents act to isolate themselves from the network through communicating with their immediate neighbours. This involves closing switches in the network within a timeframe in the order of milliseconds – thus reinforcing the idea that events with the smallest operating window are best placed as close to the source as possible.

This example demonstrates that self-organisation within a MAS can be used to trigger active reconfiguration of the electrical network instead of the communication network. However the agents themselves do not exhibit any self-organising capabilities and do not restructure themselves.

Overall the architectures presented in literature maintain a hierarchical trait, some offering centralised control and data collection while others implementing a distributed control and communication architecture. The specifics governing some of the control architecture design choices involve the visibility and availability of data and measurements, in many cases a controller will require information from several sources and therefore cannot within the same tier as the controllable entity. Certain control requirements favour a more centralised viewpoint and are less sensitive to architectural design. Therefore it would be reasonable to assume that a control and communication architecture would benefit from being able to transition from one configuration to another for the purposes of fulfilling different control requirements.

2.5 SMART GRID PROJECTS

At present, the electrical network largely operates with limited degrees of observability and controllability especially at the distribution level, which is a consequence of the historically unidirectional nature of power delivery. As emerging technologies became more prominent within the energy system in the form of energy storage, electric vehicles and distributed generation, the unidirectional model is becoming increasingly less applicable. The development and integration of these technologies in conjunction with the supporting cyber-physical infrastructure moves the power system closer to the concept of a ‘smart grid’. In many respects the term ‘smart-grid’ is often applied as an umbrella description of a wide spectrum of concepts and solutions involving new grid hardware such as energy storage, power electronics and smart meters. Smart grid concepts also extend to the inclusion of distributed control and intelligence, active customer participation schemes and forecasting techniques.

Internationally investment in smart grid research and projects has been significant as illustrated in the following table in Table 2.3 as published by the authors of [69].

Table 2.3 – Smart Grid Funding Summary

Country / Region	Forecasted Smart Grid Investments	Funding for Smart Grid Development	Smart Meter deployment
European Union	€56 billion by 2020	€384 million	45 million installed as of 2011, 240 million by 2020
United States	€238-€334 billon by 2030	€4.9 billion in 2009	8 million as of 2011, 60 million by 2020
China	€71 billion	€7.3 billion in 2009	360 million by 2020
South Korea	16.8 billion	580 million in 2009	500,000 by 2010 750,000 by 2011 24 million by 2020
Australia	n/a	253 million in 2009	2.4 million by 2013, state of Victoria
India	n/a	n/a	130 million by 2020
Brazil	n/a	143.6 million in 2009	63 million by 2020
Japan	n/a	621.3 million in 2010	n/a

The report also documents a list of 219 smart grid related projects which were in effect at the time of publication spanning both the UK and the wider EU community. These projects covered a wide range of components of the smart grid domain including grid automation, smart metering, customer behaviour and overall integrated system approaches. Within the context of the research presented in this thesis it was relevant to consider projects with a focus on investigating and deploying a control and communication architecture. This architecture could include the collection of data from smart-metered customers and the placement of either local and/or centralised controllers.

2.5.1 Smart Grid Projects in the UK

The first set of examples documenting smart grid demonstrations and deployments considers projects within the UK.

Northern Isles New Energy Solutions (NINES)

The NINES [70] project aims to develop smart grid solutions for the electrically islanded network of the Shetland Islands. Prior to the instigation of the project the islands were supplied solely by two aging convention generation plants and a single wind generation site. Potential for further wind generation capabilities was limited by network integration issues surrounding voltage levels and power flow. Adding storage capabilities, demand side response solutions and an active network management system, additional wind capacity can be added to the existing network and overall network operation can be improved.

The NINES network is a predominantly LV network serving the Shetland Islands, the network is electrically islanded from the mainland grid and prior to the project taking place was supplied by aging fossil fuel plants. The highest voltage level is 33kV, delivered via overhead cables and subsea lines when connecting between islands. It is composed of 1000 domestic customers, two conventional generation plants, one wind farm, a 1MW Battery Storage device and one 150MWh Thermal store with a 4MW boiler. The network encompasses 1,000 customers. Total customer demand peaks at 48MW and drops to a minimum of 11MW in summer – yearly average consumption of 215GWh. As per the hub network diagram the system covers 11, 11kV network branches serving the population centres. The diagram doesn't disclose where the 1,000 customers involved in the trial are located. Across the Shetland Islands there are an estimated 10,144 households[1], therefore the NINEs network encompasses approximately 10% of the total population.

Control of the network is conducted via a network management system [71], which retrieves sensor measurements from points in the network and in conjunction with internal models of different aspects of the network including demand forecasting. The active network management (ANM) control system has three core priorities: Balancing and scheduling, transient stability, and power flow and voltage management. In each instance the ANM appears to be a single centralised control unit receiving updates from various components within the system. Separate algorithms handle each of the control objectives; it is not clear whether separate control stations are used to distribute the different processes. While there are different algorithms managing the three core management conditions, they can cooperate and interact with one another and do not work completely independently. Therefore it can be suggested that the AMN system could be represented as three central agents operating in parallel and containing their own control jurisdictions, with negotiation capabilities. The remaining agents would sit within the points of generation and the smart loads within customer premises. The overall structure of the network and its control system are presented in Fig. 2.6

[1]National Records of Scotland: http://www.gro-scotland.gov.uk/files2/stats/council-area-data-sheets/shetland-islands-factsheet.pdf

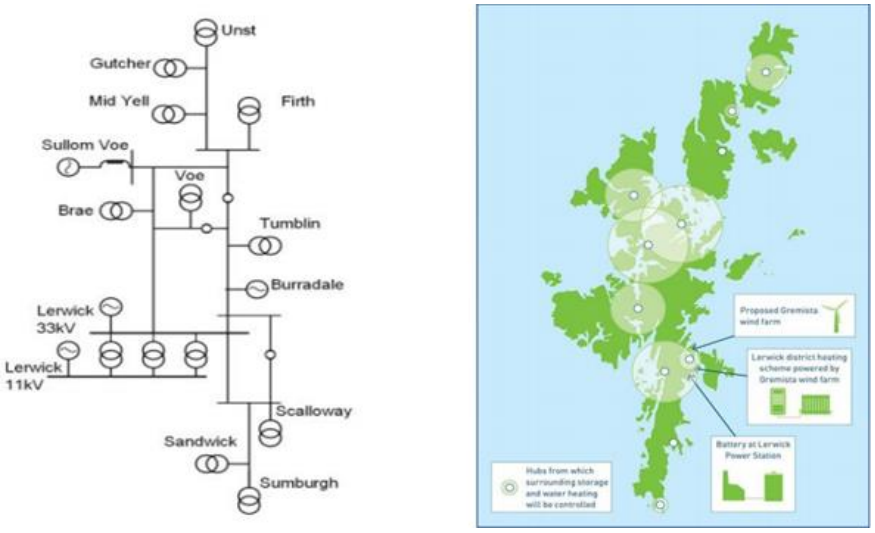


Fig. 2.6 – Electrical Diagram and Controller Locations

In terms of controllable elements within the network, the idea is that a proportion of the control will delivered through demand side response, 234 homes within the network are equipped with smart heaters which can be controlled remotely. Users can enter a series of parameters, but control signals themselves are communicated from the AMN. These signals are delivered per 15 minute intervals to the set of controllable loads to set their charging and discharging patterns for each quarter hour period. Control over storage and water heating operates in a more decentralised manner – local hubs are situated throughout the network as noted in Fig 2. The majority of the control hubs independently oversee certain physical locations, whereas the control domain of other hubs overlaps with each other. It appears that these hubs do not cover the same customer populations, for example the southernmost hub corresponds with the 11kv network branch serving Sumburgh – a population centre containing around 100 residents, whereas other hubs can contain populations closer to 1,000 residents. However given it is not stated how many participating customers exist per hub controller.

Voltage control processes operate on the basis that only pre-defined points on the network are monitored, these points represent areas which are specifically at risk of a voltage excursion. If the constraints are met at these locations, the remainder of the network will be operating within limits. Points of vulnerability are determined through prior analysis on the basis of whether adjustments to controllable devices could cause either power flow or voltage to violate the operating limits. This reduces the number of inputs requiring processing by the AMN's power flow and voltage management module. The update resolution for incoming data for this monitoring process is not stated.

As noted, the evidence tends to suggest that control is centralised with information retrieved from components. The different aspects of the control system operate under differing data transmission requirement, for example the customer demand forecasting for space and water heating processes information in groups of 100-150 customers, instead of the entire 1,000 customer population. A trial house contained 19 smart devices, each communicating 12-14 data channels at a resolution of 1-5 minutes [72]. Charging instructions are delivered in 15 minute intervals from central control, as forecasting covers 15 minute intervals.

Electricity North West – LV Network Solutions Project

The LVNS project [73] is aimed at improving monitoring across LV networks with the vision of better preparedness for future developments in domestic generation, storage and EV ownership. The project is focussed on investigating measurement approaches within the LV network to gain a greater insight into the performance of these networks. Data extracted from the monitoring approaches is then used academically as an input to network model design to investigate capabilities for projected increases in low carbon technologies.

The scope of the project contains 200 distribution substations – ground-mounted transformers and areas with PV installations were the core focus of the selection. This coverage represents a small proportion of the total ENW network containing 33,000 distribution substations. Across the 200 selected substations the scope of the project included over 1000 feeders. Some feeders connect less than 25 customers, while over 50% of the total number of feeders within the trial population connects over 50 customers.

The control and communication configuration is presented in Fig. 2.7, overall the ENW LVNS project isn't specifically concerned with performing control actions, instead primarily focussing on monitoring and data collection. The monitoring devices do not receive any form of control input from the central server, but a degree of interaction is involved in enacting the communication protocol. Therefore the monitoring devices do have command reception capabilities in addition to transmission. Modelling and analysis procedures are often conducted offline via the data files siphoned off to Manchester University, the iHost server itself does not perform any analysis or control duties – purely data collection, organising and archiving. Data collection and processing was handled by a central server, information from monitoring units transmitted using GPRS/3G mobile phone communication channels. Each monitoring device logs data in 1-10 minute intervals and transmits the information using an installed SIM card.

The central server also exported data to Manchester University for processing and further investigation. Data transmission from monitoring devices to the central collection server was conducted at 10 minute intervals – initially a sampling rate of 1 minute was implemented, but issues involving high data flow rates and volumes affecting data retrieval and storage drove a change in sampling rate. Sampling rates with a longer time interval than 10 minutes would lead to underestimating voltage impacts.

The following variables are included in the LV monitoring approach:

- RMS line to neutral voltage per phase
- Bi-directional currents per phase, and neutral currents
- Power factor per phase
- Phase angle per phase

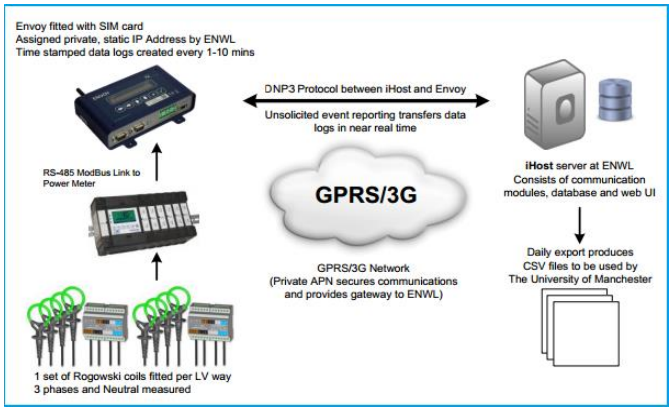


Fig. 2.7 – Communication Links in LVNS

From this information real and reactive power was calculated, along with total harmonic distortion. Each measurement packet is communicated with a sequence of header data including date, time, feature number, serial number and phase. The majority of monitoring devices were installed at the substation level, and on the LV bus at the head of a feeder. A small subset of the trialled network was monitored at the mid and end points of selected feeders – 25 feeders (50 sensors) included in the initial deployment. Plans for expansion into monitoring for 100 feeders with 200 sensors are in place. Individual customer monitoring is not in place in this scenario, customer profiles are replicated using the CREST tool, and simulated profiles were validated against a 51 customer test cell [74]. Therefore interpreting the monitoring devices in place from an agent based perspective – each feeder would be represented by three agents: one at the feeder head, plus mid and end point monitoring. Additional agents would monitor substations and transformers. In the overall

trial network this would total 3,200 agents, not taking into account duplicate points of measurement where a feeder head monitoring and substation monitoring would be under the control of a single agent.

Present measuring timescales and focusses are planned to be replaced by smart-meter data as and when widespread installation occurs.

SoLa Bristol Project

The solar Bristol project [75] [76],is of a much smaller scale than the other two examples provided thus far, with a focus on integration of PV generation and DC microgrids instead of widespread data management as noted in the LVNS example. The project fits within the Smart grid approach through integration of a cyber-physical system and demand side response approaches.

The SoLa Bristol project consists of 30 domestic customers, 5 schools and an industrial customer – therefore containing far fewer customers than the other example projects which spanned a much larger section of the host network. All customers in the trial are fitted with PV generation, battery storage and a DC connection. Multiple levels of control are applied within the project, instead of a purely centralised control solution. The lowest level of control exists within the domestic properties themselves. A combined charger inverter device is a micro-computer controlled unit aimed at managing the energy derived from the PV generation and current battery storage levels. The local controller determines the charging pattern of the connected batteries and handles exporting excess power to the grid. These controllers can be receive update commands and remotely monitored. Unlike the local smart-devices in the NINES network a degree of local automation is present with less remote intervention, this is because of the objectives of the system the SoLa approach is more customer centric whereas the NINES network is more network centric. Execution of demand side response requests is also handled locally at the customer layer via a Siemens LV connection manager device which instigates automated demand management.

Monitoring and analysis is performed at the substation level acting as a middle-agent between the customer and the central data store. The substation receives periodic measurements from the connected customers which are then processed by the LV Network Manager. The LV Network manager is another Siemens control device which determines if a demand side action is required and which customer/customers are eligible to provide

the appropriate response. The substation control level is aimed at monitoring network conditions and initiating control actions in the event that constraints are reached. Whereas the customer connection manager aims to enact the control action in the based on customer needs, battery levels and PV output.

The communication network in the project is primarily focussed around GPRS data transmission, ultimately conveying the information to a central data repository. Monitoring and analysis of received data is conducted at the substation level where each substation is in contact with a subset of the overall customer population as noted in Fig. 2.8. These substations would act in the same role as aggregation agents in the existing MAS implementation. Updates from each substation are then passed on to the central data repository - to an extent forming a similar communication structure to the clustered MAS architecture.

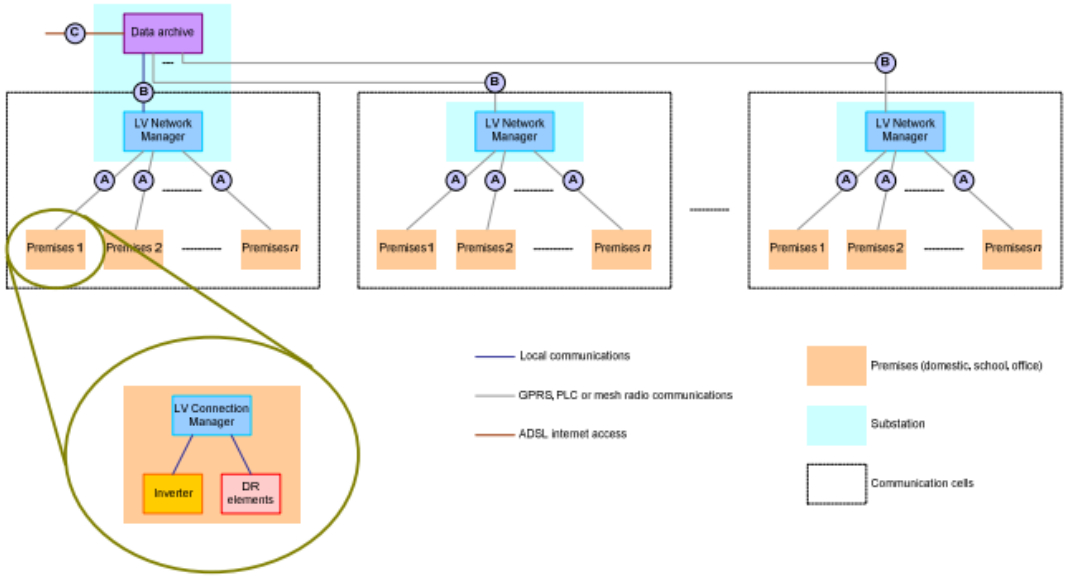


Fig. 2.8 – SoLa Bristol Communication Architecture

To avoid interference from the customer's existing internet connection and subsequent usage of the available bandwidth, each premises involved in the trial is fitted with a dedicated router to facilitate secure export of customer information. The process of data collection takes place in three stages depending on the nature of the date currently being transferred. Stage 1 is a local data store, readings are taking at one minute intervals and stored on-site, this data can be retrieved remotely if necessary and can be accessed from inside the customer home. Stage 2 is a set of periodic updates that are communicated outside of the house through the GPRS network to the data repository – these status updates

are transmitted at 15 minute intervals. The final stage relates to immediate signals which are transmitted spontaneously, in the event of a fault – information is transmitted to the substation and is passed onto the DNO.

The requirements presented in the project proposal [77] indicate that a LV network manager device – substation level controller – should allow communication with up to 32 LV connection managers. This effectively indicating that the aggregate level controller operates covers clusters of 32 customers, a clustering size that matches the customer population used per aggregate in the clustered communication architecture – presently over feeders containing 90 customers. The proposal also recommends communication thresholds between the network and connection managers.

2.5.2 Smart Grid Projects in Europe

Outside the UK, smart grid deployments remain a core component of developing and researching technologies and concepts for future grids. Due to the interconnected nature and scale of Europe the challenges facing smart-grid research can involve co-ordinating between several countries and several different organisations. As with the UK approach to smart-grid research some projects are local to one country or city with the focus on improving specific technologies. With respect to the research conducted in this thesis is was more relevant to consider cased whereby a physical deployment was discussed for the purposes of assessing the communication and control architectures involved. The first example is the Grid4EU project as discussed below.

Grid4EU

The Grid4EU project covered multiple aspects of smart grid research in the pursuit of investigating an integrated smart system with involvement from 6 different distribution system operators in as many countries (Germany, Spain, Italy, Czech Republic, France and Sweden). The core objectives of the project are outlined in the report as presented in [78] – the report also documents that an outcome of the project was the deployment of six demonstrators one in each of the six contributing countries. Across the six demonstrators a total of 275,000 customers, however due to the level at which the control and communications are applied to the network the same number of components are not involved in the communication architecture. Substations were often used as the smallest component in the architecture and therefore a small number of substations accounted for a much larger number customers for example in the German demonstration project.

In the German demonstrator project the control and communication is facilitated by a three-tier hierarchical multi-agent system, each tier of the hierarchy contains a different agent classification as illustrated by Fig. 2.9

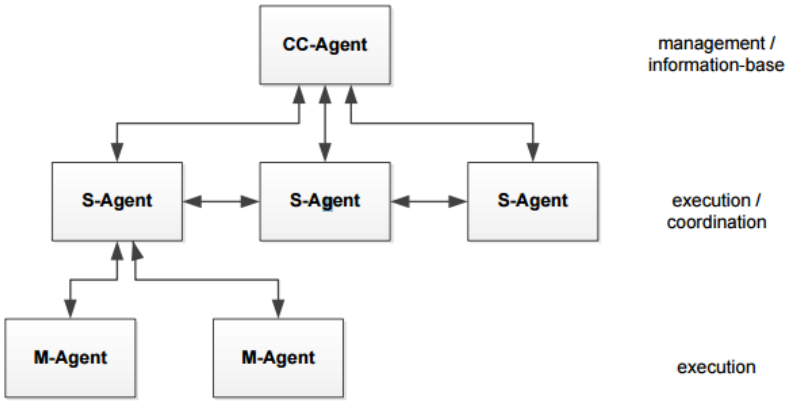


Fig. 2.9 – Grid4EU German Demonstrator Agent Hierarchy

The lower tier is populated by M-Agents, which represent non-switchable substations and therefore have not control influence on the network, in comparison to several UK projects the scope of the project is aimed at the MV level and does not consider agents at the customer level. Each of the M-Agents represents a large customer population therefore allowing a relatively small amount of agents to cover a wide geographic are. The purpose of the M-Agents is to perform measurements from the substation and other relevant information, and each of the M-Agents is associated with a single agent in the next tier. In the second tier are a series of S-Agents, these also represent substations and exist on the same electrical tier as the M-Agents but account for substations with switchable capabilities. The S-Agents are responsible for responding for collecting data from M-Agents and responding to control events, unlike the M-Agents, S-Agents can communicate with one another to co-ordinate control and share information. The top tier is occupied by the CC-Agent which acts as a gateway agent to the SCADA system, therefore it is responsible for global data collection and overall agent management. In some circumstances depending on the potential configuration of the agent architecture, an S-Agent can assume the responsibilities of the CC-Agent, therefore all of the agents involved in the architecture will be associated with substation components in the electrical network as the digital and electrical architectures take on differing structures.

Web2Energy

A second example of a European project is Web2Energy as outlined in [80], the Web2Energy project is another multinational example spanning five countries. The project focuses on three elements of smart-grid research in the form of Distribution Automation, Smart Aggregation and Smart Metering. In each of the three research criteria the over-arching theme is the analysis of the data requirements and parameters for communication. For example in the work package focussing on Smart Aggregation is concerned with retrieving data from loads, generation units and storage devices with the goal of representing the components as a virtual power plant. The data resolution for each of the elements of the virtual power plant is set at 15 minutes. As in the previous example the project contains a number of demonstrator sites to evaluate research objectives, a demonstration site in Germany is configured for the purpose of processing smart-meter data.

In terms of scale, the test network consisted of 200 smart metered customers and relied on the existing communication infrastructure to transmit smart-meter data. The communication between smart-meter and the metering infrastructure varies based on the type of customer and their connectivity methods. Some of the customers will be using their own broadband connection, this approach transmits instantaneous power data with a transition resolution of 1-2 seconds. The customers will also have access to the real-time data through the use of an in-home display station. Alternatively if the customer does not connect though a broadband connection the connectivity is provided via the 2G mobile network, customers providing data in this manner produce updates at 24hr intervals, while pricing information is passed back to the meter. The smart meter also displays information based on local measurements at 15 second intervals in addition to billing information transmitted from the supplier, the in-home displays are less sophisticated in the absence of a broadband connection. In both circumstances the communication between smart meter and the metering infrastructure is bidirectional and therefore the hardware is required to process incoming data in addition publishing updates. A third and final set of customers serves a different purpose and therefore are assigned a different communication approach, these customers are capable of performing demand side response and as a result are connected via PLC to a data concentrator. This concentrator acts as a controller and delivers the signals triggering demand side response from selected controllers.

UPGrid

A final example of a European smart-grid project featuring a deployment exercise is the UPGrid project, which is another multi-national project which indicates the challenges faced by electrical networks within Europe and their interdependence on neighbouring countries. In this case demonstration activates were completed in Sweden, Portugal, Poland and Spain as described in [81], each of which features variations in the control and communication architectures. An example architecture is represented within the Spanish demonstration project as illustrated in Fig. 2.10. As the figure demonstrates, a flat communication architecture has been implemented where data is transmitted through a secure medium to a central system. The central system is then responsible for control, power system analysis and visualisation, what is not discussed is the intermediary communication infrastructure between the layer of intelligent devices and the centralised control system. It may be the case that a series of data concentrators are implemented but if so, these components will play no role in the control and decision making process.

In a similar approach to the German demonstration component of the Grid4EU project, the smallest components within the intelligent devices layer are substations, which results in a large effective customer population while integrating a comparatively small number of agents. The demonstration project covers 190,000 customers in the city of Bilbao, as illustrated by the communication structure in Fig. 2.10 the information flow is indicated to be unidirectional and therefore the objective of the Spanish demonstrator is primarily focussed network monitoring and data collection than overall control performance.

Fig. 2.10 – Communication Architecture of the UPGrid Spanish Demonstrator Project

The other demonstrator sites within the overall UPGrid project are designed to evaluate differing requirements and therefore present with alternate communication methodologies. For example the Portuguese demonstration project is more involved with the collection and

dissemination of market pricing information as documented in [82]. The inclusion of the energy pricing data results in a communication format which involves cyclical messaging as market data is returned the customer as customer consumption information is passed to the suppliers.

2.5.3 Smart Grid Projects in the Rest of the World

Looking further afield other countries in the rest of the world are subject do differing constraints. For example Australia and the United States host grids which can cover vast geographic areas within the same county and where multi-national cooperation is less of a dominant factor. Coordination and collaboration may still be required between states/territories and operators however.

Perth Solar City

The first example is part of a national solar cities initiative in Australia in which seven urban locations across the country selected as demonstration sites for solar city concepts as documented in [84] the solar cities program aims to improve renewable energy integration, smart meter implementations and energy conservation. The programme also takes into account the social and behavioural elements of a smart grid solution for the purposes of developing demand side response objectives.

Many of the demonstration projects follow a similar format and involve the installation of PV units both domestically and on community buildings, installation of smart meters and the distribution of solar water heaters. One of the these projects was cited in Perth as documented in [83], and involved the deployment of 9,000 smart meters, the communication infrastructure for the metering scheme is presented in Fig. 2.11 and indicates a holonic architecture. Smart meters represent a Home Area Network (HAN) which is then composed of a series of smart devices responsible for heating, load control and metering. Smart meters themselves connect to the network management system in one of two manners, standard meters use the conventional 3G network via a meshed radio frequency service in the immediate vicinity of the customer area. The meshed system would offer greater reliability as each connection isn't specifically reliant on one relay or access point to the 3G network allowing data and commands to follow multiple paths to the network management system. A second variant of smart meter deployed as part of the demonstration project is a point-to-point connection, meters using this method are in direct

connection with the network management system and bypass the conventional communication infrastructure.

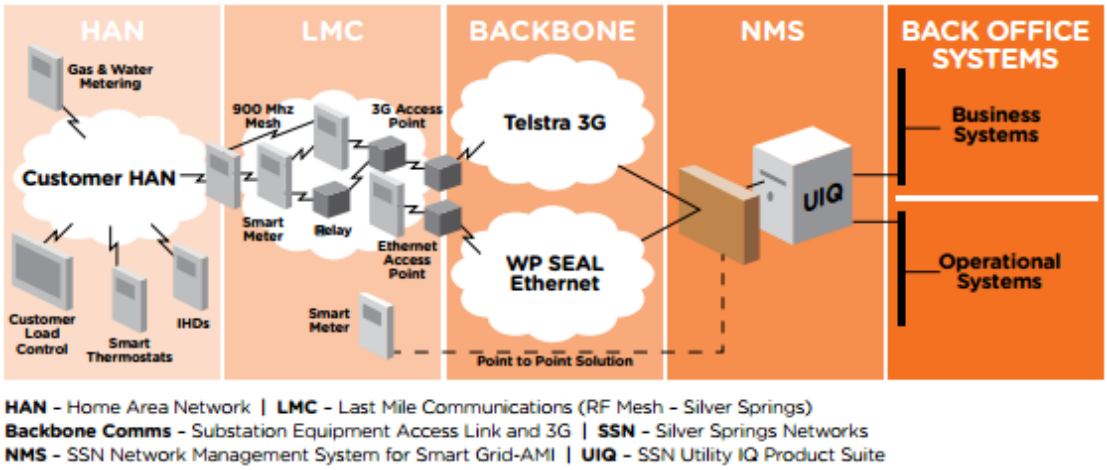


Fig. 2.11 – Perth Solar City Communication Architecture

In the Perth solar city demonstration project control is handled centrally by the network management system and additional software packages responsible for assessing business and operational conditions. In many respects this format bears similarity to the UPGrid European project whereby an advanced metering infrastructure doesn't host any of the control responsibility and all decision making processes are handled at a central location. However the Perth solar city project does cater more for applying control rather than focussing on network monitoring as the customer HAN is designed to facilitate the integration of smart loads and heating systems which can be accessed as a form of demand side response.

AEP Ohio GridSMART Demonstration Project

A further example is taken from a project implemented in the United States in the State of Ohio as documented in [85], which focuses on the delivery of four core smart grid objectives. The first of which considers the implementation of an advanced metering infrastructure and the installation of 100,000 smart meters within the scope of the project. Each of the meters are designed to accept bi-directional communication which is in line with the specifications of the meters installed in the Solar Perth Project, customer updates are transmitted to the operator and control signals are received by the set of smart meters. The motivation behind the inclusion of bi-directional communication is to achieve the second of the project objectives in the form of the provision of customer engagement for the purposes of demand side management. Demand side management schemes are a recurring concept in the series of smart grid project examples in conjunction with a smart

metering infrastructure. The third of the core objectives is to increase network automation through increasing the embedded intelligence within the network to automate assets and provide network reconfiguration capabilities. A final objective of the project is to optimise voltage control procedures through the integration of new control algorithms and approaches.

While the core objectives previously outlined are consistent with properties of the other smart grid research projects, the AEP Ohio project specifically outlines procedures for dealing with security incidents. Initially a central monitoring system contains software which can detect meter tampering for the purposes of electricity theft. However in addition to the prevention of meter tampering the project also includes scope for the development of a suite of cyber-security processes – this was introduced through more thorough testing and evaluations of technologies involved in the demonstration project. The technologies were subject to a threat analysis and penetration testing procedures – such an approach had not been explicitly highlighted any of the preceding examples on smart grid deployment. In addition to the pre-emptive security analysis performed on the components, collaboration between project contributors and additional agencies added scope for further analysis of an active system. American Electrical Power (AEP) as one of the lead contributors was involved in the development of a cyber-security operations centre (CSOC) as documented in [86]. The core objective of the centre is to recognise the threat posed by cyber-attacks to the smart grid and facilitate collaboration between energy companies in the US to exchange information relating to attacks and mitigation strategies.

2.6 VULNERABILITY TO CYBER THREATS

In a system which is increasingly reliant on ICT infrastructure, there are additional points of vulnerability and therefore potential weaknesses which a malicious user may wish to exploit. Given that several industrial control systems presently offer very little in the way of security adding increased connectivity between controllers and other nodes in a unified cyber-physical system does increase the accessibility of these unprotected systems. Therefore the role of threat resistance and attack response become increasingly important in the role of managing the power system. The following figure in Fig. 2.12 taken from the Internet Security Threat Report [87] illustrates the rise in detected vulnerabilities within industrial control systems across an increasing range of vendors.

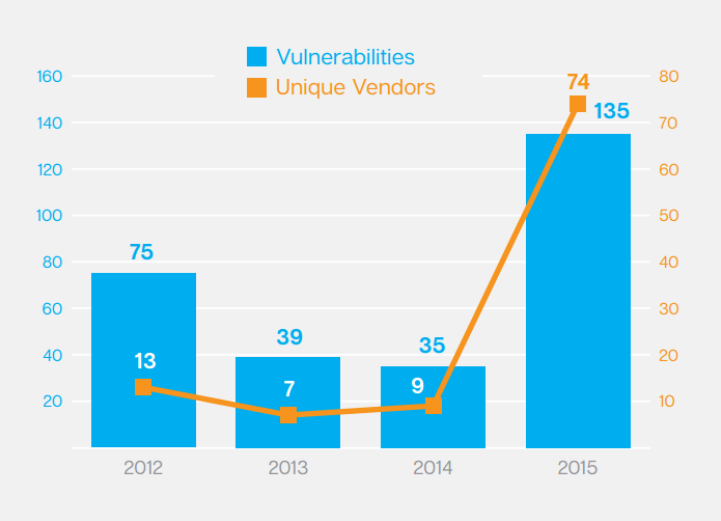


Fig. 2.12 - Increase in ICS Vulnerabilities

There are no defined models or definitive attack processes because an adversary will not adhere to a set of particular guidelines and instead will develop the attack based on the desired goals of the adversary and the resources available. It is often considered that a potential attacker with an unlimited amount time, skill and resources will be able to break through any potential security measure circumventing all defensive measures. Therefore it is not feasible to develop and build a completely impenetrable system. The following sub-sections outline a series of potential attack methodologies and the potential impacts; this is not an exhaustive list of strategies and consequences but demonstrates the range of vulnerabilities and consequences of an attack event.

2.6.1 False Data Attacks

One attack approach is in the form of false data, whereby an attacker injects information into the system, adding false data into a network can take place in a number of formats, from misrepresenting sensor data to trick state estimation systems [88] or falsify network topologies [89]. False data based attacks can trigger controllers to make incorrect decisions resulting in economic losses and operational issues [90].

False data based attacks have been demonstrated to pose a threat to a range of elements of network operation. For example research has examined the case for the vulnerability of state estimation tools to this attack format: [90], [88], [91], whereby false information is accepted by the estimator bypassing bad data filtration processes. These processes are also able to identify and remove malicious measurements which injected into the system, a false data attack aims present data which passes through error detection solutions and is then

used in the process of performing state estimation. An inaccurate state estimation compromised though data injection could lead to incorrect decision making or reduced awareness of component failure. Another attack strategy involving the use of a false data injection approach involves misleading the control centre via falsifying network topology information [89]. In this attack approach the attacker exploits the lack of authentication between terminals and a control centre; as a result the attacker can convince the control system that the network is operating under an alternate topology. Therefore making it easier to conceal network stresses and prevent control actions being initiated to relieve those stresses. The authors of [89] indicate that the lack of authentication is a result of the volume of legacy devices and communication equipment active in the power system – it is these devices which indicate the scope of the problem of defending a large scale cyber-physical infrastructure against attackers.

2.6.2 Malware Attacks

A second approach is to infect devices within the system with malicious software designed to either compromise a device such that it can be used as a launch platform for an attack on another part of the network or modify/disrupt the functionality of the host device. Malware based attacks such as Havex [92], and BlackEnergy [93] have been traced and monitored by the Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) in the US. This illustrates the validity of the threats posed by malware [30] to the power system, the use of malware may be one component in a multi-faceted attack on the network - for example an attack on the Ukrainian power system as described in [30] included several attack vectors one of which was the delivery of malware – using a variant of the BlackEnergy malware previously noted. The same attack also included the distribution of manipulated Microsoft office documents with embedded malware to allow the help the attackers gain control of the system.

2.6.3 Denial of Service (DoS) Attacks

DoS attacks involve transmitting a large volume of traffic at a target or targets with the view of interrupting or disabling the service delivered by the target. Like malware based attacks, DoS of Distributed DoS events can be conducted using bespoke software or tools available online. As per the information provided in [87] such an attack can be purchased for as little as $10 per day indicating that in many respects a DoS based attack method may be the most universally accessible to a wider range of potential attackers as the required

technical knowledge is lower. The scale and scope of a DDoS attack can vary considerably depending on the connectivity of the target and the resources of the attacker – for example recent DDoS event targeting the BBC as reported in [94] reached a peak attack traffic transmission rate of 602Gbps. In the context of a smart grid network scenario this level of attack traffic can be considered unlikely given the capabilities of the control and monitoring equipment involved. While the authors of [13] indicate that the present cyber-physical network has little in the way of defensive approaches in the face of a denial of service attack other than purchasing additional bandwidth.

While large scale denial of service events can prove to be destructive and have the potential to grind digital systems to a halt operating over several days or weeks – they are easier to detect than some other of the more latent attack formats. However a form of the denial of service approach relies on targeted bursts of low-rate data over a period of time aiming to interfere with the operation of a system when it is most vulnerable. This as referenced in [95] and [96] is called a shrew attack and aims to be more efficient and require greater awareness of the target system.

2.6.4 Social Engineering

One of the most difficult weaknesses to counteract from a technical standpoint is the use of social engineering methodologies which rather than launching an attack against the hardware or software of the infrastructure targets the human controllers. Social engineering aims to breech security protocols through convincing operators that access is needed or through manipulating the user to unknowingly divulge access credentials or rights to a potential attacker. In itself social engineering is unlikely make up the destructive element of an attack and serves more as an intermediary step on the path to perpetrating the attack. Using the example of the attack on the Ukrainian power network as referenced in [30] a social engineering mechanism called spear-phishing was employed to steal access credentials to the system which in turn would have made the installation of remote access tools easier and hand control of the system over to the adversary.

2.7 CONCLUSIONS

The examination of the literature and network deployments there general consensus is that a hierarchical approach to the design of the control and communication architecture structures is the most effective method. However the research is not unified on the structure

beyond the hierarchical concept, citing differing levels numbers of data aggregation levels, and distribution of controllers within the architecture. The concept of self-organisation was barely recognised within the source literature and indicating that the architecture considered within the research are intended to remain static, likely with additional reinforcement to ensure they can withstand a wider range of communication traffic. Therefore there is definite scope to explore a range of architecture designs using the information extracted from existing literature and determine whether there are performance gains to be exploited from the design of the communication architecture. Furthermore there is additional scope to consider if the absence of a thorough discussion of self-organisation within the presented examples is an oversight or whether a well-designed static configuration is capable of producing all of the relevant performance characteristics.

Additionally the communication structures present in the deployment examples are not necessarily aiming to solely deliver control signals to components within the network, the ENW-LVNS example mainly oriented around the provision of data collection and information management. Both these factors will have to be integrated into the agent architecture in order to evaluate their effectiveness. The structure of the networks examined will serve as a basis for the underlying model in terms of feeder length, customer population distribution and architecture configurations. Although the SoLa Bristol project presents a situation whereby only 32 customers can be effectively connected to a local controller, predicted advances in communication technology and computational power would suggest that this upper boundary could be exceeded in the future. Therefore these limits will be removed when considering the architectures presented in the following chapter, certain configurations will consist of smaller customer clusters than others but there will be a higher threshold. If the investigation determines that self-organisation is a viable method of defining the control and communication architecture, these limits will be useful in determining how many connection requests a controller or aggregator will accept. Additionally the deployment scenarios present cases with a relatively low sampling rate or degree of observability. For example the ENW-LVNS case includes three points of voltage sampling per feeder, at the top, tail and centre of the line, with data transmitted from sensors at up to 10 minute intervals. The planned investigation intends to increase the communication rate and coverage within the agent population, partly to be in line with potential future technologies but also to test the architectures more thoroughly with a more intense communication demand.

From a development stand point multi-agent systems are commonplace within power system and specifically microgrid research and therefore demonstrating that such an approach is an appropriate base for the investigation into architecture designs. The investigation into the differing agent platforms has concluded that using JADE the development tool is an appropriate technique for exploring the different architectures and for implementing a series of agents which can represent the technologies involved in a sample network model.

Finally research considering the impact and potential vulnerabilities arising from cyber-threats indicates that these network events should be considered when developing smart grid architecture. This is because a cyber-attack may be directed at the control and communication technologies with a goal of disrupting and interfering with the delivery of control signals.

Overall the research has indicated that there is a case for developing multi-agent architectures while individual scenarios and configurations prefer variations on a hierarchical theme. Therefore the following investigation will consider a series of architecture configurations in isolation to evaluate their relative performance, this will determine the potential plausibility of introducing self-organisation as a method of accessing ensuring continued system operation in the event that the network requirements change, or an attack event is present.

Chapter 3: Multi-Agent Architectures for Voltage Control

3.1 INTRODUCTION

Chapter two outlined the structural concepts used within the smart grid research domain featuring levels of decentralised control and hierarchical communication structures. From this information a series of architecture configurations were subsequently developed drawing inspiration from existing research. A total of 16 control and communication architectures were constructed within the JADE agent platform, each of which was then examined in the presence of a voltage deviation event present on each feeder, such that as additional agents were added to the scenario, the scale of the control requirements also increased. Therefore raising not only the amount of data recovered through customer demand, DG generation and voltage monitoring data streams but also through the number of requests for control and the resulting control signals.

This investigation considered the performance of the individual architectures in isolation, to establish whether there was a notable difference between the capabilities of the selected configurations. Each of the architectures was assessed for its performance in respect to conventional operation in the form of performing the voltage control objective and maintaining data flow between the tiers of the architecture, and under the influence of an attack event. In addition to performing the control objective the architectures were also assessed under the presence of cyber-threat which was designed to be a result of compromised smart-meters failing to respond to control signals.

This chapter outlines the selected electrical network upon which will be under the jurisdiction of the agent population and documents the control process embedded within the set of agents with respect to calculating voltage magnitude and providing voltage control. The chapter also contains the structures of the four core communication architectures developed, and the nature of the agents involved within those architecture detailing the roles and responsibilities of each component. Finally a series of test criteria are considered for evaluating the performance of the differing architectures and the series of results generated in response to those criteria.

3.2 TEST NETWORK

The selected network model used for the testing process was based on a series of radial LV feeders; a section of LV network was selected due to the distribution network being subject to the introduction of smart-metering. Furthermore it is a part of the electrical system which is presently under-observed and therefore was an appropriate tier in the power systems

hierarchy to install agents. Additionally the literature surrounding control and communication architectures mostly involved the distribution network and smart-metering infrastructures. This network model is outlined in the following diagram in Fig. 3..

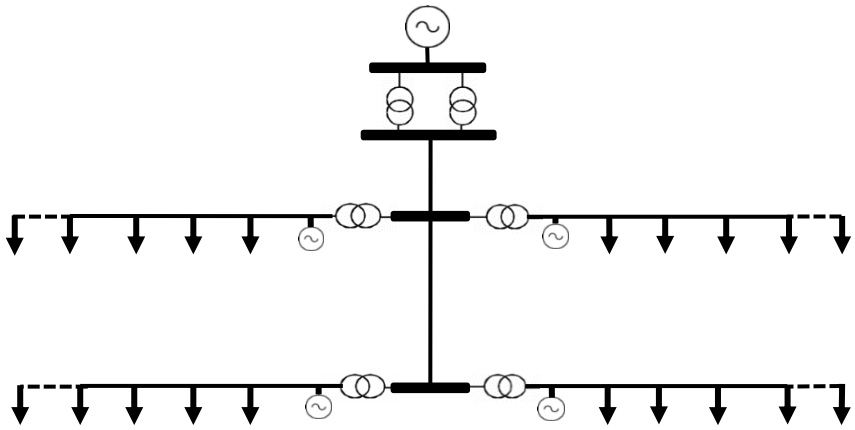


Fig. 3.1 – Network Diagram

The network existed as a series of embedded variables stored as agent knowledge within the agent population. Each agent was supplied with information describing impedances to its neighbours, using data extracted from [97] and presented in Table 3.. The variables and calculation processes were then verified against a Simulink model of the same network configuration to ensure that the agent community modelled an electrical network. The network was composed of several radial feeders, each serving 90 customers, the number of customers, length of cable and customer density parameters were in line with feeder represented within the LV Network Solutions Project as documented by the authors of [73]. Population increases were achieved through appending additional branches to the network spine up to a total of 18 feeders serving 1620 domestic customers.

Table 3.1 – Network Parameters

Parameter	*Value*
Generation	63kW per DG
Customer Demand	External demand profile
Branch Separation	500m
Customer Separation	10m
Feeder Population	90 Customers
Number of Feeders	6-18 (on branch pairings)
Customer Population	540-1620

11kV Conductor	
Resistance	0.164Ω/km
Reactance	0.08Ω/km
400V Conductor	
Resistance	0.32Ω/km
Reactance	0.075Ω/km

Each of the customers was supplied with a demand profile which was extracted from Customer-Led Network Revolution (CLNR) data files available online [98]. The number of customers per feeder and the selected load profiles were configured such that a voltage deviation would be present on each feeder, affecting customers towards the end of the feeder.

3.2.1 Voltage Calculation

The voltage calculation process was triggered by the observer agent at three second intervals – the calculation propagates through the network. Each active agent in the hierarchy was informed of the voltage at the preceding bus a value for transmitted power; the agent knows the impedance of the connecting conductor. Using this information the agent was then able to calculate the voltage drop between itself and the preceding agent as a means of calculating the voltage at the bus the agent represented. The voltage drop calculation is presented in the following equation (1).

$$(1) \quad (V_r - V_s) = {}^{RP}/_{V_r} + {}^{XQ}/_{V_r}$$

Where V_s and V_r are sending and receiving voltages respectively. All values were converted into per-unit under the following base values (2-4).

$$(2) \quad S_{base} = 100MVA$$

$$(3) \quad Zbase11kV = \frac{(1.1x10^4)^2}{1x10^8} = 1.21\Omega$$

$$(4) \quad Zbase400V = \frac{400^2}{1x10^8} = 0.0016\Omega$$

The voltage calculation is then completed iteratively for each pair of electrically connected nodes throughout the overall agent network. This allows each of the customer agents to be aware of the voltage at their corresponding bus and thus determine if any control actions need to be taken.

3.2.2 Recognising and Tracking a Developing Excursion Event

In order to prevent the controller from intervening too quickly, events were tracked before being responded to. When responding to a voltage violation a controller would instruct customers to shed load, reducing demand in order to raise the voltage level. It was important to ensure that the voltage deviation event was a persistent event rather than a transient one before shedding customer load.

Therefore a stand-off period of time was introduced, initiated at the when the excursion event is first detected. If the event persists to the point that it lasts longer than the stand-off period – then intervention action is taken. The stand-off time was set to five minutes, such that transient voltage excursions do not trigger a control response. The length of this period is shorter than the voltage regulations as per the UK grid code – continuous operation up to 15 minutes [99] - but it exceeds length of the longest voltage sag definition (events between 0.5 cycles and 60s) as defined in IEEE standard 1159 [100] outlined in the following table - Table 3.2.

Table 3.2 – Voltage Sag Classification Table

Voltage Sag/Dip Classification	Timeframe
Instantaneous	$^1/_2$ cycle – 30 cycles (10-600ms at 50Hz)
Momentary	30 cycles – 3seconds (600ms -3s at 50Hz)
Temporary	3 – 60s

Each customer was responsible for the monitoring the length of any voltage deviation it observed, this responsibility was not passed onto the control layer for two reasons. Firstly this reduced the communication overhead as each deviation did not have to be reported to a controller, and only those events which exceeded the stand-off time were communicated to a controller. Secondly it removed the necessity of the controller to maintain a series of timers devoted to each incident, as each customer contained its own event timer which is triggered as soon as the calculated voltage drops below 0.94 per unit.

3.2.3 Performing Control Actions

Once a voltage issue had been detected and determined to be a persistent problem the customer detecting the event then began an interaction with the controller. This interaction involved sending a control request to the controller and waiting for the controller to issue commands to customers on the same feeder as the deviation. If the architecture was operating with the highest level of decentralisation – i.e. customer led control – the first stage would be bypassed by the fact that the control functions and the source of the deviation were the same agent. Control was achieved through demand side response – modifying customer demand rather than modifying generation output. In the network model, 50% of the customer population were deemed to be controllable and would accept control commands to reduce demand by 700W. Once a controller received a request it would then select an initial set of customers and send query messages to those customers

to check if they were controllable. If any of the selected targets indicates that they were a controllable customer, the controller would send the instruction to shed load in order to reduce the under-voltage situation. This interaction is presented in the following diagram.

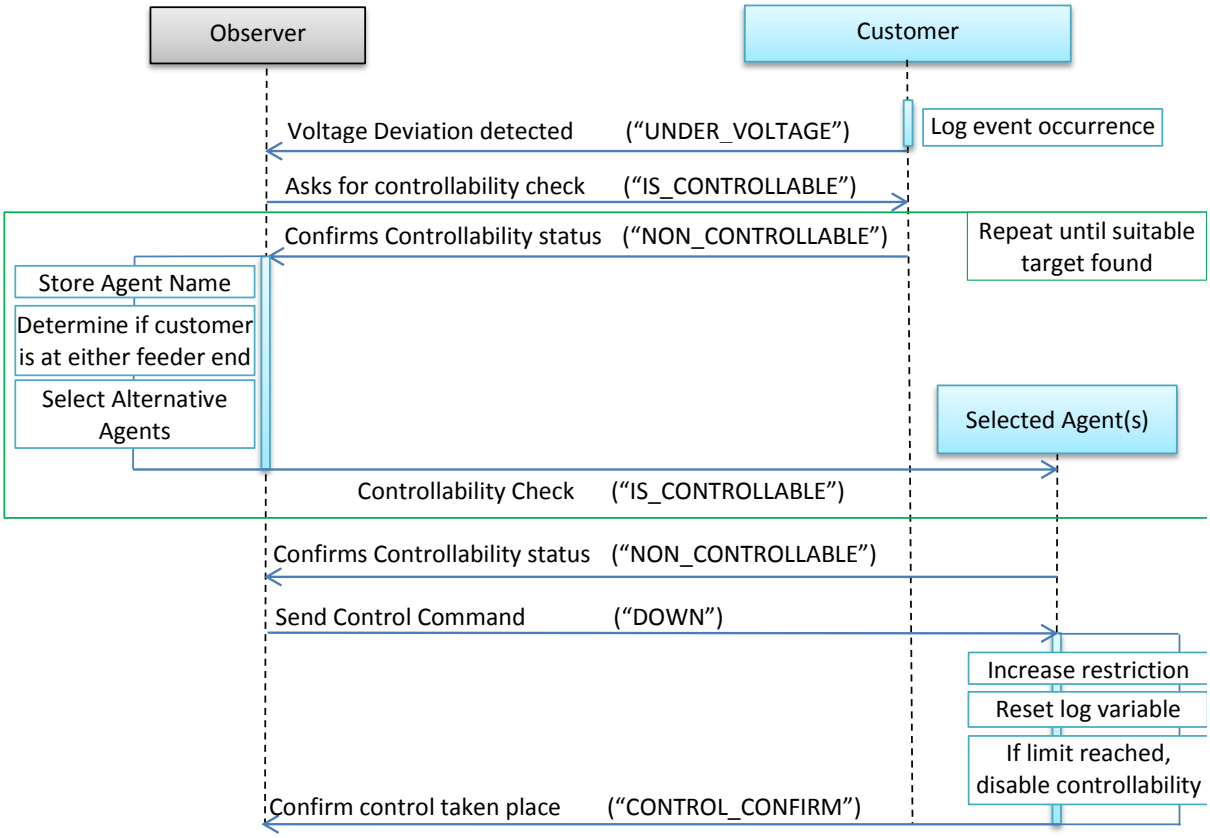


Fig. 3.2 – Communication flow between agents during control

The load shedding restrictions were left in place until the voltage recovers past a safe value whereby the controller would then issue a set of commands to lift all restrictions. This safe value was set at 0.96 P.U. and would only be accepted when received from the agent at the end of the feeder, using data from customers closer to the 11kV-400V transformer at the head of the feeder would result in the restrictions being lifted too soon and thus reducing the effectiveness of the process on reducing the voltage deviation.

3.3 ARCHITECTURE DESIGNS

Four initial organisational structures were implemented for testing purposes based on the core concept of a hierarchical design which was prominent in literature. Within these architectures the location of the control functions could be varied for differing levels of decentralisation. For example a fully centralised control mechanism would involve customer agents transmitting control requests to the central observer agent. Whereas a

highly decentralised control approach involved customer agents transmitting control signals to neighbouring customers without involving any other tier in the hierarchy. As previously indicated the issue of increasing network scale and agent population numbers was performed through adding further feeders to the end of the main network spine. The four architecture designs implemented are presented as follows.

3.3.1 Base Architecture

The first of the architectures which was developed was a base architecture, in this configuration a single aggregate agent is allocated per feeder. This aggregate agent was responsible for the collection of demand and generation information from its feeder and triggering the sequence of voltage calculations which would then iterate down the length of the feeder. If the aggregate was not responsible for customers on feeders connected to the furthest end of the central network spine – the aggregate would also trigger voltage connections to the corresponding aggregate in the subsequent network branch.

The control mechanisms involved are dependent on the level of decentralisation – customers would contact the controller directly without following the communication route involved with transmitting customer data. For example if centralised control was in place, a customer would send a control request to the observer agent instead of that message being routed to the observer via the aggregation tier. This architecture is presented in the following figure Fig. 3.3.

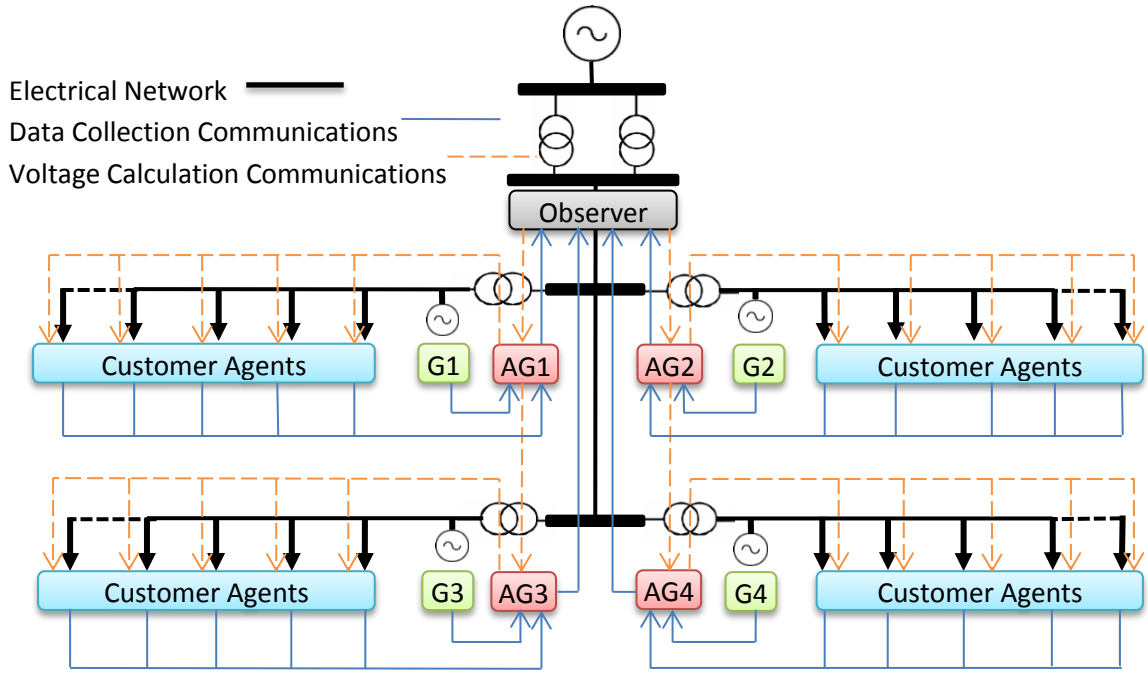


Fig. 3.3 – Base Architecture Diagram

3.3.2 Clustered Architecture

The clustered architecture increased the number of aggregation agents per feeder, and therefore reduced the communicative load at each of the aggregation agents. To perform the voltage calculation sequence one of the aggregates on each of the feeders was selected to act as a cluster-head agent, which meant it was responsible for starting the set of sequential equations which were then passed between customers along the length of the feeder. The cluster head agent was supplied with the demand information collected from the rest of the aggregates on the feeder such that it could perform an initial calculation. Each cluster head agent would then supply demand and voltage information any cluster head agents electrically downstream.

The cluster head agent was on the same hierarchical tier as the other aggregates, but was the only aggregate associated with a feeder which would report to the observer agent. The overall objective of the architecture was to reduce the congestion at the aggregation layer as this was a key bottleneck area within the structure. The topology and communication structure is presented in the following figure: Fig. 3.4.

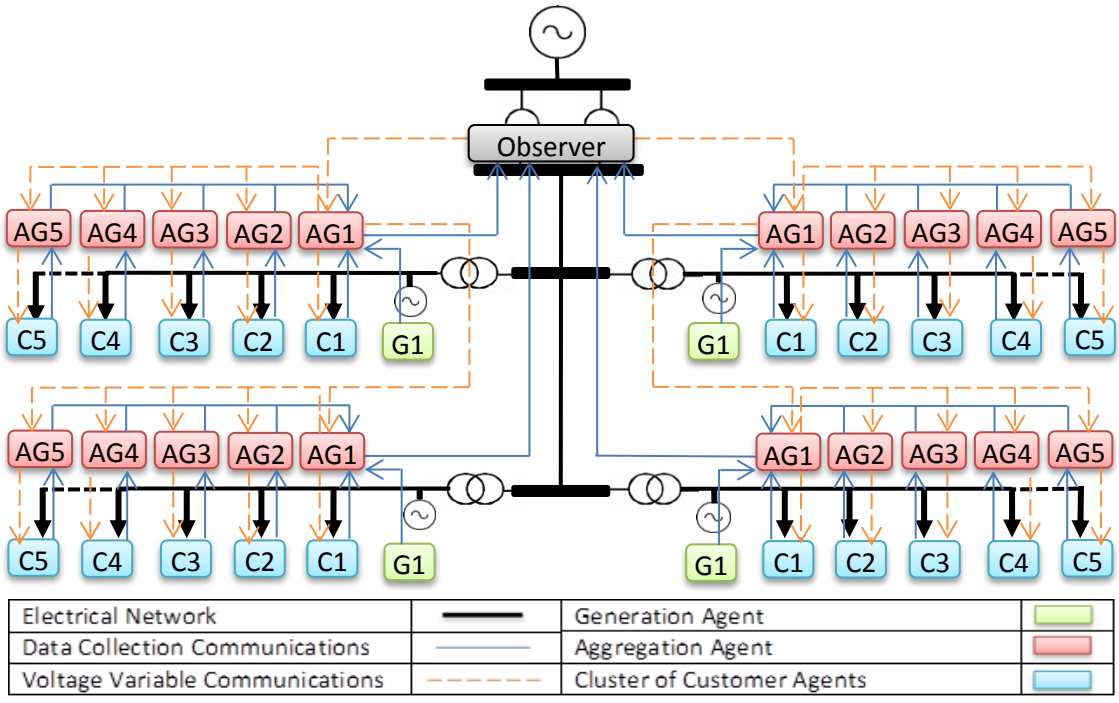


Fig. 3.4 – Clustered Architecture Diagram

3.3.3 Tiered Architecture

The third of the core architectures was built to accommodate the role of a cluster head agent as a separate tier within the hierarchy. Its secondary function was to place a buffer stage

between the main aggregation layer and the central observer agent, such that any congestion issues were shielded from the central agent by an addition aggregation layer.

With the exception of handling upstream power values for individual customers the upper tier aggregate performs the same core duties as it did in the other configurations. It received demand information from the set of aggregates on the feeder and relays that onto the observer to gain a total picture of the network. The generation agent on the feeder also reports directly to the upper tier agent, as in the clustered architecture it would communicate with the cluster-head agent. The following figure in Fig. 3.5 presents the structure of the architecure.

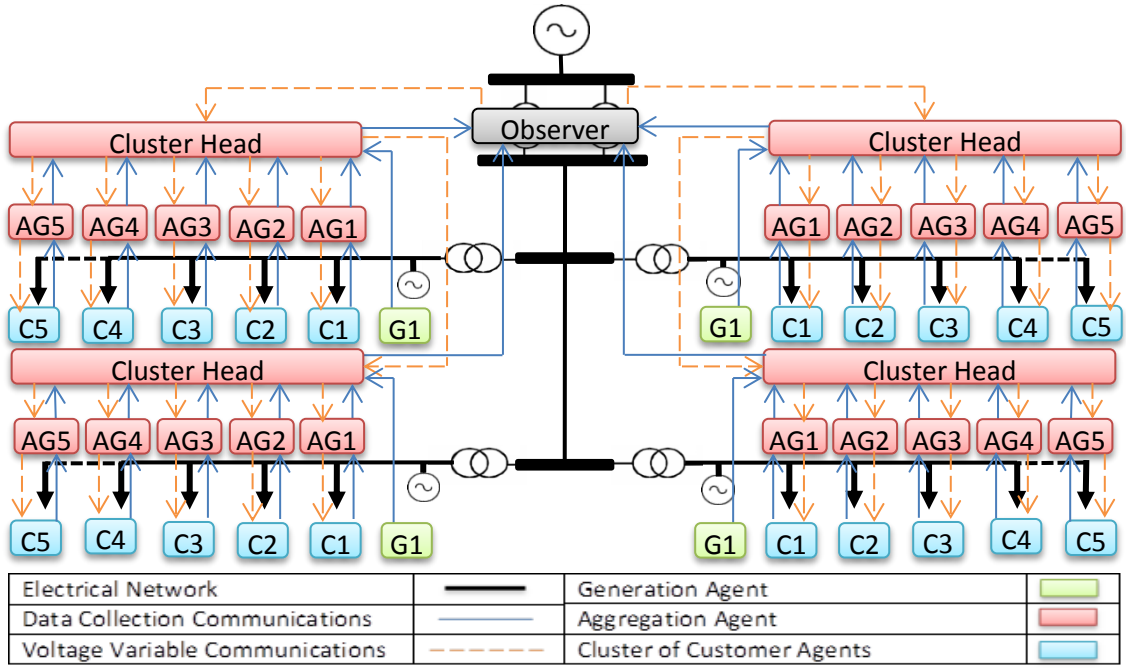


Fig. 3.5 – Tiered Architecture Diagram

3.3.4 Disaggregated Architecture

The final developed conversion was the disaggregated architecture, which involves removing all of the dedicated aggregation agents from the architecture. Instead the generation agent assumed the responsibility of the sole aggregation point for each of the feeders. This means that fewer total agents are active on the platform in comparison to the other designs, but increases the potential for congestion. There were no dedicated aggregate agents present within the architecture and therefore fewer potential layers at which the control process could be installed. The topology for this architecture is presented in the following figure in Fig. 3.6.

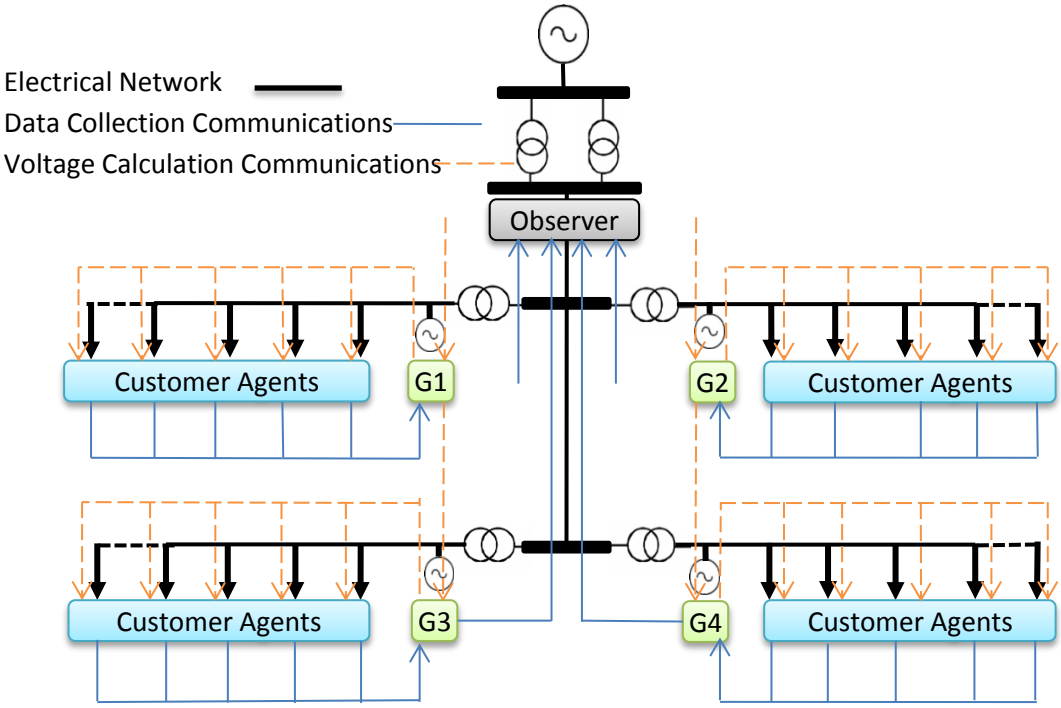


Fig. 3.6 – Disaggregated Architecture

3.3.5 Control Levels

Within each of the previously mentioned architectures, multiple levels of control decentralisation, whereby each of the agent classifications present has the capability of receiving control requests and disseminating commands. A completely centralised approach uses the Observer agent as a controller and therefore all customer agents report any detected voltage deviations to the central point. At the opposite end of the scale the customer agents themselves request control actions from their neighbours in response to a deviation event.

Taking the four architectures and the control levels in account a total of 16 permutations of control and communication architecture combination were investigated and these architectures were then examined under three escalating population scales. Each scale extended the electrical network through adding additional feeders to the central network spine, the agent architecture would be applied to the extra feeders as per the structure of each of the architectures previously introduced. The additional feeders are identical to the previous ones such that the distribution of loads remained balanced between scales and such that the absolute volume of agents and messaging became the dominant variable.

3.3.6 Security Considerations

The four presented architectures are subject to a series differing performances advantages and disadvantages with respect to being resilient to a cyber-attack event, additionally the alternate control approaches would also create a differing set of properties. The nature of the impact of an attack event is related to the objective of the attacker and the attack methodology employed. One objective may be to disrupt the control ability of the controller, and therefore control requests from the customer layer are either ignored or responded to incorrectly. Alternatively the target of the attack could be to gain access to information stored within an agent, such as demand profiles and customer details.

Any architecture which was running under a centralised control mechanism contained a central point of failure at the observer, which if compromised by an adversary would have significant consequences for the controllability of all of the controllable customers. However it is also reasonable to assume that a central control room would have the most sophisticated defences in order to reduce the chances of an intrusion. However if the architecture was under the control of a set of aggregation agents, the impact of the attack would be influenced more by the topology of the agent architecture. The Base Architecture contained one aggregate per feeder and therefore a compromised aggregate would only impact the feeder with which the aggregate was associated. Whereas a clustered architecture required more aggregate controllers to be compromised before the entire feeder would lose controllability, therefore in addition to enabling a reduction in communicative load the clustered architecture also offered greater robustness.

If the control was located at the customer tier, it would take several customers to be compromised before a degree of control loss could be achieved. This is because multiple customers are required to perform a control action and therefore if a small number of customers had their controllability disabled or manipulated overall controllability will not be affected. However in the presence of smart-meters as customer agents, it is these agents would be considered the most accessible to a potential adversary, unlike the majority of the agent types within the architecture it is the smart-meters which provide the easiest physical access and therefore could be tampered with by an attacker.

3.4 AGENT SPECIFICATION

In the development of the architectures presented in the previous section test configuration a core agent population needed to be established representing a generic distribution network composed of several different agent types. These agent types are presented below.

3.4.1 Customer Agents

The customer agents formed the core population of any of the implemented Multi-Agent architectures and were designed to represent customer smart-meters. The function of the customer agents was to maintain local load profiles and relay periodic updates to the aggregation layer such that a global picture of demand could be drawn. In addition, the customer agent was also responsible for handling information pertinent to voltage calculations and passing that information onto neighbouring agents. The final function of the customer agent was to perform local voltage monitoring, once the customer has performed a voltage calculation as discussed in section 0, the agent assessed the result. If this result was outside of the recommended +10%/-6% voltage limits, the customer agent would start the event timer. In the event that the deviation event proved to be persistent – the customer would then be responsible for contacting a controller to alert it to the developing problem. If the architecture was operating with the highest level of control decentralisation, the issue would be processed internally; otherwise the customer would receive control signals from other tiers within the hierarchy. A percentage of the customer population was configured to perform demand side response as a control mechanism; the customer agent was responsible for receiving those control signals and applying the relevant action to the local load profile.

3.4.2 Generation Agents

The generator agent class operated in much the same way that customer agents did, it had an internal profile which defined the output of the generator. This information was then passed up the agent hierarchy to the aggregation layer, in the event that a configuration without a dedicated aggregate population is in place, the generation agents assumed the responsibility of becoming the aggregation points. All updates were passed directly to the observer agent, after the generator agent would receive information from the customer layer. Another potential role for the generator agent was as a controller. If a persistent deviation was detected, control requests would be received by the generation agent on the same feeder as the deviation event. The concept of generation control within the context of the

presented research referred to the fact that the generation agents would transmit control signals to controllable customers, rather than being able to influence the generator output.

3.4.3 Aggregation Agents

The Aggregation agents are placed in a tier above the customer layer, this tier was predominantly used for the purposes of data collection from customer and generation agents. Depending on the architecture in use the aggregates would either forward aggregated data onto the observer if no other aggregates were present on a feeder. Otherwise, the aggregate agent would forward updates from the generation and customer population to the cluster-head aggregate instead of the observer. The aggregation agent could also be used as a local controller, only responding to the control requests of the customer population within its catchment area. This catchment area was dependant on the number of aggregates in operation per feeder; this may span the length of one feeder or a subset of agents on that feeder.

3.4.4 Cluster Head Agents

A cluster head agent acted as an enhanced aggregation agent, whereby it contained all the core functionality of the other aggregation agents within architecture it also performed further duties. Cluster-heads were used when multiple aggregates were assigned to a single feeder, all demand and generation information collected by the other aggregates on the feeder were passed to the cluster head. This was because the cluster-head agent needed to know the overall demand of the feeder such that it could perform the first voltage calculation, before reiterating the information to the first customer. In addition to triggering voltage calculations along a feeder the cluster head would also pass voltage information to the next cluster head agent downstream. All demand and generation information collated by the cluster head agent was passed up to the observer agent.

3.4.5 Observer Agent

The observer agent was the central entirety in each of the architecture designs; it represented a central control room or server which is supplied with the global data from the network. All demand and generation updates were transmitted to the observer which built an overall picture of the network. The observer agent could also act as a central controller, if used as a controller all control requests from the customer population would be received by the observer and it would be responsible for disseminating the control signals. This approach creates the potential for the observer to make decisions on a global level – and

may have wider control applications but it also created the risk of a control request bottleneck at the top of the architecture. Furthermore it also represented a single point of failure; therefore if the observer was no longer able to perform the control responsibility as a result of an attack event or failure, controllability would be completely lost throughout the architecture.

3.4.6 Error Generator Agent

The final agent that was included within the population was the error generation agent; this agent was not part of the core architectures from a communication and configuration perspective. The role of the agent was to deliver the signals required to simulate failure or attack events within the architecture when instructed to do so. The error generator would transmit an instruction to a controller to begin performing anomalously in the form of rejecting control requests or responding with incorrect commands. When there were not attack or failure events taking place, the error generation agent served no function and did not interfere with the general running of any architecture design.

3.5 PERFORMANCE CRITERIA

To assess the performance of the different configurations presented in the previous section of this chapter a series of comparative metrics were introduced to extract information from the agent population. These metrics illustrated the impact of architectural design choices on the core control objective of solving a voltage deviation in addition to the impact on the levels of computational load the system experienced. A further purpose of using multiple metrics was to determine if the competing architectures favoured certain properties or had side effects from increasing control performance. These metrics are as follows:

3.5.1 Congestion

The congestion metric focused on the number of messages which were trapped in the message queues maintained by each of the agents. When an agent within the Jade platform received a message from another agent it was temporarily stored in the message queue, once it had been read by behaviour within that agent it was then removed from the queue. If the agent was subject to a large number of incoming messages, for example in the case of the aggregation layer receiving periodic updates from an agent population, it reached the point where more messages were being received than being read. As a result the message queue began to build and this caused data congestion, by measuring the amount of messages

waiting in this queue it was possible to gauge the level of congestion that particular agent was facing. As previously noted the issue of congestion was largely a problem of the aggregation layer due to the nature of the responsibilities it faced.

3.5.2 Reactivity

Reactivity referred to the ability of the architecture to respond to events taking place within the architecture, it acted as a measure of the speed of detection and response to certain messages transmitted by the customer layer of the architecture. To evaluate the reactivity, each of the customers was instructed to transmit a message to its controller which contained information about a fault condition. Two stages of reactivity were then considered, the first was the time difference between the initial fault message being transmitted, and it being detected by the control layer. In the eventuality that the architecture was operated with customer level control then detection times were not recorded, this was because the source of the message and the detector were the same agent. The second reactivity consideration focussed on the controller responding to the fault message and transmitting a command signal back to the customer agent. This test evaluated the distance between controller and customer and its impact on reactivity, but also the degree with which message congestion compromised the ability to detect and reply to key messages.

3.5.3 Message Efficiency

Message Efficiency dealt with the matching the number of messages transmitted in total with the number of messages received - to determine the percentage of messages that reached the ideal destination. Messages are only counted if they are processed by the recipient agent rather than being captured by the message queue – therefore messages which were trapped in a queue were not counted as they were not processed. As congestion and data flow increased the likelihood that a message would not be processed increased and therefore lowered the message efficiency of the overall system. For many messages such as demand and generation updates, the issue of message efficiency was not necessarily a significant problem – it played a far greater role in the transmission of critical messages. Control requests reporting voltage deviations required action from the recipient and therefore in a system with lower message efficiency it became less likely that those messages would be acted upon and therefore reduced the controllability of the system.

3.5.4 Control Performance

Control performance related to the ability of the controllers to solve the voltage deviation – each simulation was provided with a set of profiles which contained an under-voltage incident which required solving. A customer would report an under-voltage incident once it lasted for more than 5 minutes, confirming it as a persistent issue. This report was made to the controller for the corresponding section of the network who in turn will issue control commands to resolve the issue. An architecture configuration performing strongly would be able to complete this interaction faster than one under the strain of communication problems and therefore solve the voltage deviation in a shorter period of time. As the voltage performance is a key indicator of the electrical performance of the system it was important document the impact the cyber layer had on the physical network properties.

3.5.5 Robustness

Further research considered examining the set of architectures from the perspective of robustness and the ability to perform the control actions under the pressure of component failure or cyber-attack.

During the investigation the selected method of failure focussed on the control action, enacting a potential avenue for a cyber-attack. Those customers who were designated as 'infected' would still perform monitoring, and detection of voltage deviations –they would also still also accept control signals from a controller. The influence of the infection was that it changed how the customer agent responded to an incoming control signal – instead of performing the load shedding action requested, the agent increased the demand. This could be through activating a heat-pump, electric vehicle charging station or other smart-loads within the premises – an infected agent increases demand by 1500W – which risks intensifying the voltage deviation and nullifying any control attempts. The infection only targeted controllable customers, which amounted to a maximum 50% of the total customer population. To examine the survivability of this form of attack, different quantities of infected agents were activated per feeder.

Even through the selection of a single attack strategy there is an immense quantity of potential permutations of attack location, strength, spread pattern, duration, and scope. An attack could be limited to a single feeder, or be network wide – each feeder population could be affected differently depending on the differing vulnerabilities of the installed smart-meters or monitoring devices. This list of attack vectors is then further expanded

through variations in the customer population – a network with 1620 customers has considerably more avenues of attack than one with 540 customers for example. Therefore it was important to narrow the scope of the investigation an effective worst case attack scenario was selected – wherein the infection affected each feeder symmetrically and one agent population size was selected of 540 customer agents. Adding further feeders to the investigation would not change the nature of the problem as each feeder would respond to the attack in an identical manner. The only difference would be in the magnitude of the voltage deviations further from the grid connection, as more customers experienced the load increase, the voltage drop in the network spine would be greater and therefore feeders at the far end of the network would experience lower voltages at the start of the feeder.

Each of the sixteen control and communication architecture combinations was supplied with the same customer profile calibrated to create a single voltage deviation event, and the first 1000 seconds of the voltage profile was analysed. This section of runtime would comfortably cover a period of pre-deviation operation plus the event and recovery time, under normal circumstances. As the number of infected agents per feeder was increased up to the maximum 45 (50% of a feeder's population) the length of time taken to correct a voltage deviation should increase. The upper limit of 1000 seconds was used to compare levels of infection in which the deviation cannot be corrected to prevent the length of the deviation being defined by the simulation runtime. The voltage profile for each customer was processed to extract the duration and magnitude of any deviation event experienced. The total number of deviation events and excursions was also recorded; where an event refers to any period of time the voltage drops below 0.94pu whereas excursions only consider events lasting more than 300 seconds. Eight levels of infection were tested per configuration – starting with control tests of 0 infected agents, and no control and increasing the number of infected agents to 5, 10, 20, 30, 40, and full infection of 45 customer agents per feeder.

Several metrics were recorded for each of the customers within the overall architecture, as each customer agent regularly updates a CSV output file containing data documenting time-stamped voltages to reconstruct an individual profile per user. These profiles were then dissected via a Matlab script to isolate the periods of under-voltage, counting the number of occurrences in addition to retrieving the duration and magnitude of each event. From each profile the following information was recorded:

- **Max Voltage**
- **Min Voltage**
- **Mean Voltage**
- **Number of Deviation Events**
- **Number of Events Exceeding the 300s waiting period**
- **Max Deviation Event Length –** Timed from the first instance of voltage dropping below 0.94, until recovering above limits
- **Total Under-Voltage Time –** Total time spent below 0.94pu
- **Average Deviation Length –** Taken using all deviation events, not just those exceeding the 300s stand off period.

3.6 RESULTS

Two series of investigations were conducted into the performance of the range of control and communication architectures. The first of which considered the operational variables of the network considering control performance, alongside communication metrics. The second investigation was focussed on operating the set of architectures in the presence of an external attack which compromised agents at the customer layer.

3.6.1 Operational Performance Results

Each of the 16 configurations was examined against four criteria as published in by the author of this thesis in [101]. The results demonstrated that the 16 different architecture combinations displayed variances in performances that indicated that no single configuration out-performed the others across the range of agent population scales and differing performance metrics as presented in the following figure in Fig. 3.7 from the paper.

	Performance Metric	Control Performance			Congestion			Reactivity			Message Efficiency			Shading Legend
	Customer Population	540	1080	1620	540	1080	1620	540	1080	1620	540	1080	1620	
Control and Communication Architecture Combination	Base + Central	6	5	5	9	8	14	9	7	6	6	8	7	1st
	Base + Aggregation	10	14	14	8	6	13	16	16	13	4	3	2	2nd
	Base + Generation	7	11	11	15	15	12	8	11	10	15	14	15	3rd
	Base + Customer	5	10	9	13	16	2	11	9	12	13	13	16	4th
	Clustered + Central	15	6	7	16	2	10	7	14	15	1	5	4	5th
	Clustered + Aggregation	2	4	4	5	5	7	5	4	3	8	9	5	6th
	Clustered + Generation	1	2	1	4	9	9	3	5	9	10	10	9	7th
	Clustered + Customer	3	1	3	12	13	3	2	2	4	16	11	12	8th
	Tiered + Central	8	9	8	14	10	8	4	3	2	2	12	11	9th
	Tiered + Upper Aggregation	14	7	10	2	4	6	15	12	11	9	15	8	10th
	Tiered + Lower Aggregation	4	3	2	1	1	5	1	1	1	12	2	10	11th
	Tiered + Generation	12	8	13	7	12	11	6	6	5	11	16	13	12th
	Tiered + Customer	9	13	12	10	11	4	10	8	8	14	7	1	13th
	Disaggregated + Central	16	12	6	11	3	16	14	15	16	3	1	3	14th
	Disaggregated + Generation	11	15	15	3	7	15	13	13	14	5	4	6	15th
	Disaggregated + Customer	13	16	16	6	14	1	12	10	7	7	6	14	16th

Fig. 3.7 – Performance Summary Table

The figure presents the relative rankings for each of the different architectures for four of the performance indicators previously introduced. Illustrating that some common design trends delivered preferential performance but did not deliver the strongest performance across all categories – for example configurations with increased aggregation capacity are able to respond faster to changes in the network and operated with lower message congestion. However the same configurations scored worse in the context of message efficiency as a result of a larger number of aggregation points in the communication architecture increasing the chances that messages were not be received. Based on the results it was determined that the architecture would benefit from not being fixed to a single configuration; this is due to the fact that differing architecture designs exhibited differing properties. Therefore the data indicated that the presence of a self-organising architecture with the capability to transition between states would be beneficial in terms of accessing specific performance advantages.

3.6.2 Robustness results

The second result set documents the series of robustness tests which demonstrated the network performance under the pressure of an attack event within which customer agents were compromised. The attack targeted the smart-meters represented by the customer layer of the architecture an affected how those customers responded to control commands. The profiles presented in Fig. 3.8 illustrate the control process in the absence of an attack event.

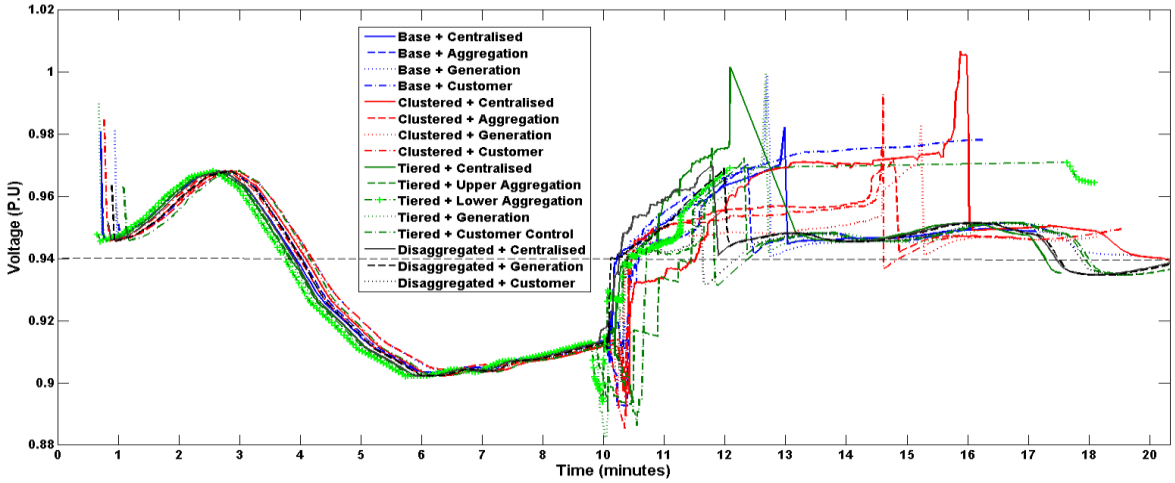


Fig. 3.8 – Voltage Profiles Without Infected agents

The profiles presented in the figure were taken from the customer at the farthest point of the network from the grid connection point at the tail of the 6th feeder – customer number 540. As the infection was distributed symmetrically and the demand/generation data was identical for each customer the selection of the final customer for the purposes of constructing the figures does not impact on the validity of the results and presents the feeder at the greater risk of a voltage deviation.

The figure illustrates that in each case the control solution was able to respond to the voltage problem and raise the voltage levels above the minimum threshold. The performance difference in the control processes between architectures does confirm the information posted in the previous table as the configurations based on the disaggregated communication architecture struggle in terms of control performance than the other formats. The following figure in Fig. 3.9 presents the same voltage profiles where 45 customer agents per feeder were infected with the malware and all control commands were reversed.

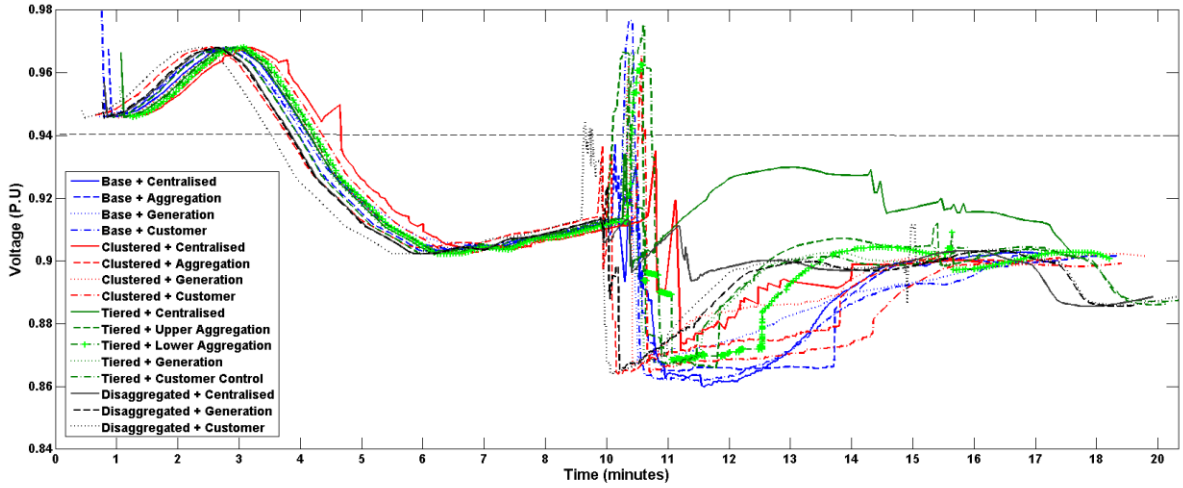


Fig. 3.9 – Voltage Profiles with 45 Infected Customers per Feeder

The figure illustrates that the different architectures responded to the presence of an infection differently, those architectures featuring a larger aggregation population were able to weather the attack event better than those with at most one aggregate per feeder. Furthermore the centralised control approaches also performed favourably against the more decentralised options due to the increased distance between the point of decision making and the component under control. In the context of achieving control without the presence of an infection this distance can be considered detrimental as the increase in the number of communication hops increases the risk of a delay. However while the attack or infection is in place, that control distance acts as insulation and defers some of the impacts of the attack.

To evaluate the comparative performances between the architectures in the presence of increasing attack intensity each configuration was ranked – as in the research conducted on an attack free scenario in the published paper [101]. The following figures illustrate how those rankings fluctuated with respect to different performance metrics extracted from the voltage profiles.

	NC	0	5	10	20	30	40	45
Base + Centralised	2	6	7	1	15	15	7	11
Base + Aggregation	5	5	1	9	3	3	6	12
Base + Generation	3	9	10	15	10	1	16	16
Base + Customer	4	10	5	7	16	16	11	15
Clustered + Centralised	8	3	8	3	7	7	2	1
Clustered + Aggregation	1	11	6	2	8	12	9	8
Clustered + Generation	6	16	11	8	11	6	15	5
Clustered + Customer	7	14	15	16	9	2	13	14
Tiered + Centralised	9	2	2	5	12	13	3	2
Tiered + Upper Aggregation	11	7	13	12	13	9	5	9
Tiered + Lower Aggregation	15	15	14	11	1	8	10	13
Tiered + Generation	10	13	12	14	5	4	8	4
Tiered + Customer	14	12	16	13	14	14	12	6
Disaggregated + Centralised	12	1	4	6	2	11	1	3
Disaggregated + Generation	13	4	3	4	4	5	4	7
Disaggregated + Customer	16	8	9	10	6	10	14	10

Fig. 3.10 – Average Minimum Voltage Ranking Table

The first figure presented in Fig. 3.10 demonstrates the variance in relative performance across the different architecture combinations and the different levels of infection with respect to the average minimum voltage per feeder. The results indicated that different levels of attack intensity are better served by alternate architectures – for example a clustered communication structure operating with generation agents providing control signals is the lowest ranked architecture when no customer agents are infected. However the same configuration is ranked 5th when faced with the most severe level of infection as 50% of the customer population are ineffectively responding to control signals.

The second of the assessment metrics considered the average duration of all under-voltage events recorded across the entire customer population, the rankings for this metric are presented in the following figure in Fig. 3.11. In contrast to the data on minimum voltage the manner in which specific architectures either climbed up or descended down the rankings was more progressive and exhibited fewer sharp changes in relative performance. This indicated a state which presented with a degree of graceful degradation, not all configurations followed this pattern – the disaggregated architectures in particular observed significant performance loss when processing the more severe attack formats. These results also indicated that a single configuration was not advantageous when under pressure from the different attack severities. Disaggregated architectures performed strongly when the attack severity was low, but the clustered architecture became more prominent when the attack was stronger.

	NC	0	5	10	20	30	40	45
Base + Centralised	14	9	11	10	11	8	13	13
Base + Aggregation	10	2	1	5	8	14	16	16
Base + Generation	9	15	16	16	16	13	5	2
Base + Customer	8	11	12	11	10	6	8	10
Clustered + Centralised	6	12	10	13	12	12	10	6
Clustered + Aggregation	2	13	13	9	14	15	9	5
Clustered + Generation	16	16	14	15	13	16	3	4
Clustered + Customer	7	10	15	14	15	5	2	1
Tiered + Centralised	15	5	7	7	2	4	12	14
Tiered + Upper Aggregation	1	8	6	3	5	7	15	8
Tiered + Lower Aggregation	5	14	9	12	4	3	6	3
Tiered + Generation	11	7	4	8	9	10	7	7
Tiered + Customer	3	6	5	4	6	1	1	9
Disaggregated + Centralised	12	3	8	6	1	9	11	11
Disaggregated + Generation	13	1	2	1	3	11	14	15
Disaggregated + Customer	4	4	3	2	7	2	4	12

Fig. 3.11 – Average Total Under-Voltage Time per Affected Customer

Finally the data illustrated in Fig. 3.12 presents the comparative rankings for the number of affected customers – a customer was considered to be affected if it encounters a voltage deviation which lasts for more than five minutes during the simulation. Higher ranked performers contained the smallest number of affected customers, in the smaller attack configurations several configurations had the same number of affected customers and were therefore ranked equally.

	NC	0	5	10	20	30	40	45
Base + Centralised	6	4	4	4	6	9	13	9
Base + Aggregation	2	2	2	2	3	13	15	14
Base + Generation	6	15	4	4	2	16	10	12
Base + Customer	6	4	4	4	6	11	9	14
Clustered + Centralised	6	4	4	4	6	1	6	1
Clustered + Aggregation	1	1	1	1	1	6	12	6
Clustered + Generation	6	16	4	4	6	12	11	3
Clustered + Customer	6	4	4	4	6	8	5	9
Tiered + Centralised	6	4	4	4	6	1	8	1
Tiered + Upper Aggregation	6	4	4	4	6	15	15	13
Tiered + Lower Aggregation	4	4	4	4	6	5	4	7
Tiered + Generation	4	3	3	3	5	14	7	5
Tiered + Customer	6	4	4	4	6	7	1	9
Disaggregated + Centralised	6	4	4	4	6	1	2	4
Disaggregated + Generation	6	4	4	4	3	4	14	8
Disaggregated + Customer	2	4	4	4	6	9	3	16

Fig. 3.12 – Number of Affected Customers Ranking Chart

The results echo the information presented in the earlier voltage profiles as configurations with a more decentralised control approach were ranked highly when faced with a low volume of infected customers. Whereas when exposed to a larger infected population the

centralised control architectures were ranked higher – demonstrating that the control distance between component and controller reduced the impact of the attack. It should be noted that under the larger scale attack formats, all feeders experienced a voltage deviation which could not be corrected as the number of infected agents outnumbered those providing correct control and therefore counteracted any valid control decisions.

3.7 CONCLUSION

Across the set of results in both the case of conventional operation and the presence of a cyber threat there is no dominant architecture configuration than can be universally described as being the strongest performer and the most resilient. This is more relevant in terms of dealing with an attack event where the impact of the same attack varies across the set of architectures and as that attack intensifies those architectures which initially ranked higher start to record a lower placing. The amount of factors influencing performance within the agent structure itself indicates that a static structure is not completely fit for purpose as the vision for future networks aims to incorporate greater flexibility and controllability. As a result it would be valuable to devise a system whereby the architecture in use would be able to change and adapt itself in response to the state of system health – either in the form of communication issues or in the form of an attack. These changes would allow the architecture restructure to deliver more effective performance or becomes more resilient to an ongoing attack event. A self-organising architecture would be able to deliver the greater resistance to an ongoing threat as it will have the capacity to isolate members of the agent population or replace agents under attack and increase the number of available controllers or data collection points to improve the performance if necessary.

Therefore on the basis of this result set the following course of action was to investigate the usage and approaches of self-organising architectures in a variety of applications where a large number of individual components are required to interact. The investigation aimed to examine tools and techniques involved with developing self-organising systems such that the implemented architecture could be informed by proven examples.

In addition to the development of the self-organising architecture it was also relevant to introduce a flexible test environment within which the voltage calculations can be conducted outside of the agent population. This was considered to avoid architectural transitions from disturbing the voltage calculation process within the agent population, the

connections within the architecture were restructured as a result of a transition event – access to the required variables to perform voltage calculations would be compromised. For example if a transition involved removing an aggregate – one which was performing a cluster head role – an additional overhead would have been required to reconnect the iterative voltage calculations with its replacement. Therefore separating the load flow mechanism from the self-organising architecture prevented similar situations from developing. This requires the addition of an external load flow engine operating over a network model of the test system and supplied with the most recent demand and generation data from the agent population.

Furthermore the data attained through the investigation demonstrated that suitable focus for the self-organising architecture was the presence of a cyber threat. This focus was based on the impacts of the attack event being more severe than those experienced in those architectures which were ranked poorly for the other performance metrics. Failing to intervene in the event of a cyber threat would be more costly than improving individual metrics when the architecture is not under attack. It was still important to consider monitoring these performance metrics in the eventuality that a severe disturbance took place which would compromise the control objective, but in terms of trigger events an attack based scenario was more relevant.

3.8 SUMMARY

This chapter documented a series of investigations on static architectures which considered differing performance metrics and the stability of the control system under an attack event. Initially the core architecture designs were described and the components of the test system were discussed including the method for voltage magnitude calculation. The results illustrated that there was a clear need for the introduction of a self-organising architecture with the ability to perform suitable transition events which aimed to improve performance. This was deduced by the variance in performance of the individual static architectures with and without the presence of an attack event taking place. In conclusion it was determined that the threat of a cyber-attack was a more relevant trigger factor for the implementation of self-organisation.

Chapter 4: Self-Organising Systems

4.1 INTRODUCTION

After conducting preliminary investigations concerning the comparative performances of different static architecture designs both in terms of withstanding a cyber-attack event and performing voltage control, the results indicated that there was scope for the development of a self-organising architecture. The self-organising architecture needed to fulfil the control and monitoring requirements contained within the static configurations but also the ability to recognise that a transitional event was needed. Therefore performance monitoring techniques were also an important consideration when examining mechanisms for implementing the system. Both communication level and electrical information was needed in terms of building the self-monitoring aspect of the self-organising architecture, information which then informed a decision making element before a transition would be made.

To develop the relevant functions and abilities additional literature needed to be interrogated for the purposes of examining tools, techniques and applications of existing research in the field of self-organisation. The investigation aimed to discover methods which could be adapted for an implementation in the smart grid domain, and to establish the requirements for a self-organising architecture. A range of different research areas were explored including wireless communications, sensor and vehicle networks because self-organisation architectures within power systems is not a field which had been documented extensively in the literature. Therefore it was relevant to consider alternative domains wherein communication and monitoring are conducted over wider geographical areas or involving multiple agents.

This chapter considers the core concepts involved within self-organisation and the desirable deliverables for implementing such a system. Following this the chapter continues on to consider differing applications of self-organisation across different research domains with a view of exploring the different approaches and considerations made for providing self-organisation. Furthermore the chapter introduces examples whereby self-organisation has been investigated for certain functions within the power system research domain. Finally conclusions are drawn on the basis of the material examined, and gaps in the research are identified where the implementation of a self-organising architecture may be of value, supporting the research presented in this thesis.

.

4.2 SELF-ORGANISING CONCEPTS

Over-arching research in self-organisation defines it as "the mechanism or the process enabling a system to change its organization without explicit external command during its execution time" [38]. Therefore all processes involved with data collection, performance monitoring and ultimately decision making are to take place within the agents involved in the system. Therefore no external influences or commands are involved in changing the organisational structure. Secondly the definition states that the system must remain in operation during a transition and cannot be offline until such time that the re-organisation process has been completed. As a result self-organising systems also need to be agile, as documented by the authors of [37]. The authors of [38] go on to outline differentiations between strong and weak self-organisation which is based around the location and method of decision making with respect to performing an architecture transition. A strong self-organised system is defined as one that is composed purely of decentralised decision making and architecture transitions – at no point is a central entity or agent involved in dictating to the network which structure to transition into. A weak system on the other hand, conducts self-organisation through a central entity which is present within the host architecture; hybrid systems perform different self-organising tasks at differing locations in the structure of the system.

Several properties and characteristics are described within the paper, included in the list are the following notable properties:

Endogenous global order – This states that the system requires the capability to organise itself into a stable state from start-up without external assistance. Therefore agents within the system form their own connections and interactions with other agents in the population depending on their roles and responsibilities.

Simple Local Rules – This involves simplifying the rule base on the local scale and the decisions involved with determining whether or not a transition is required. Therefore introducing local decision making in the case of assessing performance metrics – those which are within limits are not pursued further while those which exceed thresholds are subject to further analysis. This characteristic maps directly with the requirement for simplicity discussed in [37], wherein reducing the complexity of the individual components within the self-organised system increases the potential for that system to withstand the impacts of scale and improve agility.

Dissipation – A requirement for dissipation refers to the ability for the system to achieve a state of equilibrium, therefore following a transition event or an initial start-up stage the system does not form an unstable state, or one which is unable to perform the functions of the system. This prevents a necessity for continual re-organisation and restructuring as a result of developing unstable system states. This is also reflected in the requirement for stability as presented in [37] – whereby following initial configuration or a transition event the structure of the network settles and remains stable.

Self-Maintenance – A final property refers to a need for self-healing and the ability to remain functional in the event of failure – from a purely software perspective this would include repairing agents and agent mobility. But in a system where hardware failures could also be the cause of faults in the network, self-maintenance could be implemented through redundancy in both hardware and software.

An alternative focus on self-organisation is presented by the authors of [102], who present a scenario based around a fully centralised approach, which sits outside of the system under control. This approach does not contain a decentralised method of performing self-organisation and therefore would be considered to be a weak configuration by the authors of [38]. It is indicated that the system under control (SuOC) exhibits decentralised properties for the purposes of its own operation, but the processes involved in self-organisation would be handled by a pair of components as illustrated in Fig. 4.**Error! Reference source not found.**.

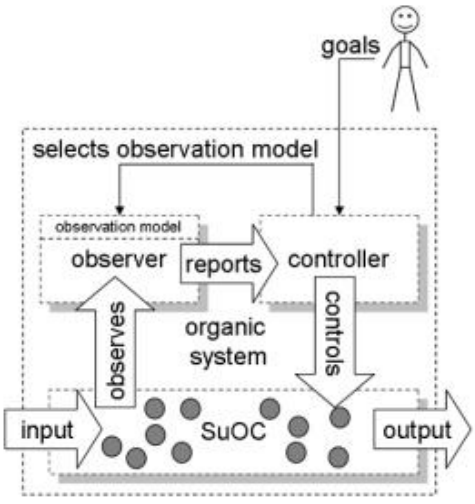


Fig. 4.1 – Observer/Controller Architecture

The idea that both the observer and the controller sit outside of the SuOC is in contradiction to the principle that the SuOC is to conduct self-organisation without external influences or control. The observer component of the self-organising mechanism is tasked with aggregation of data retrieved from the SuOC; this data includes performance indicators from individual components and global indicators. This data is then processed and aggregated before being passed to the controller. After receiving an aggregated set of data detailing the system state, the controller has the ability to make a series of potential decisions based on three potential objectives:

1. to influence the system such that a desired emergent behaviour appears,
2. to disrupt an undesired emergent behaviour as quickly and efficiently as possible
3. Construct the system in a way such that no undesired emergent behaviour can develop.

To achieve one or more of these objectives, the controller can apply one or more of the following control processes.

Influencing Local Rules: This is a command to a particular agent within the SuOC, where the controller instructs it to modify its internal behaviour. This could entail a variety of different commands such as modifying thresholds on monitored performance metrics or changing how the targeted agent interacts with other members of the community.

Influencing Structure: This is a more widespread action, changing the global behaviour of the network, which may involve introducing behaviour changes to all of the individuals within the global system or the topology of the network through modifying the number of individuals within the population. Alternatively this could entail restructuring the connections between entities within the SuOC.

Influencing the Environment: Influencing the environment requires the agents to have the ability to modify the physical nature of the system those agents are responsible for. For example, a smart grid self-organising architecture could restructure the topology of the electrical network in response to a fault condition. Not all applications have the potential to influence the host environment to a significant degree, but control decisions and actions taken by the agents involved can have impact on elements of the host system. Therefore influencing the environment may be a consequence of actions taken in fulfilment of the other control process or a standalone decision.

Selecting one of these approaches is based on a set of monitored parameters within the system under control – if these parameters exceed their defined threshold values, or the observer detects anomalous behaviour a decision is triggered. Each set of parameter states is mapped to a desired action such that the controller is aware which decision to make given the overall state of the system. However this mapping is not an exhaustive set of potential system states and behaviours and therefore, the controller needs to be able to perform decision making in the absence of a pre-selected control process. If the outcome of the control process is beneficial, the decision would then be recorded and used for future reference.

4.3 APPLICATIONS

Current research regarding the application of self-organising architectures within the power systems and smart grid domain is not extensive, and therefore to build a wider picture of the tools and techniques used it was relevant to examine literature from a range of research domains. Multiple domains have considered the concept of self-organisation and the specific use of self-organising architectures; these include communication networks, sensor networks and vehicle network management systems. Some applications consider the use of self-organising processes for the purposes of solving a particular control problem, others are focussed on the structure of the communication architecture and restructure for data handling. A range of research domains was examined for the purposes of interrogating the tools and techniques implemented, the aim was to explore a collection of methods which could be adapted and combined to develop the novel self-organising architecture.

4.3.1 Communication Networks

One of the areas of investigation examined was communication networks, where signal strength and failures can force the network to reconfigure in order to maintain connectivity between nodes. Communication networks were considered a suitable source of self-organising literature due to the parallels between the requirements and properties of a communication network and the static architectures examined in the previous chapter. A node in the context of a communication network exhibited similar properties to an agent within the smart grid architecture because they shared common roles and responsibilities. Both concepts consisted of a network of connected entities which were involved in the sharing of information and communicating between one another. While a communication network operated in the presence of differing challenges, including node mobility and

dynamic communication loads the overall structures and self-organising mechanisms were transferrable. Several communication networks employ self-organising principles as a method for handling node mobility, particularly in wireless networks [103] and mobile phone networks [104].

One example of a communication network exhibiting self-organising properties considers the data collection issues surrounding the deployment of smart-meters as described by the authors of [105]. In this example the work considers the smart-meter nodes as data producers alone and doesn't consider the implications surrounding control decisions, and the transmission of control signals, because the research is focussed on the communications element. From a self-organising architecture perspective the work examines an initial configuration stage whereby smart-meters form connections with data collection points or concentrators on a higher tier within the communication hierarchy. The use of concentrators can be considered to be analogous to aggregation agents, although the data collection points do not contain the capacity for performing control actions and are solely used to retrieve information transmitted by the smart-meters.

To form the connection between the smart-meter and the collection point, a two stage process is required. The first stage involves each smart-meter ranking the connectivity between itself and each of the available concentrators; the ranking is a comparison of the communicative distance between the two entities, and the available connection capacity of the concentrator. This process allows each smart-meter to build a list of concentrators ordered by connection preference, and to form a connection between to the most favourable concentrator. The second stage takes the form of monitoring the newly formed connection, in this stage the smart-meter listens for elongated silences from the concentrator. A long silence suggests the concentrator is no longer available and the smart-meter reverts back to stage one to select a new connection. This process indicated that two key properties needed to be integrated into the self-organising architecture, firstly the necessity to be able to analyse the potential connection options between the customer and aggregation layer before forming a connection, and secondly the ability to monitor the stability of that connection.

A second example examines the differing perspectives for a hierarchically structured radio communication network [106] based upon the 3GPP management reference model [107]. The authors base the research on the SEMAFOUR project vision as presented in [108]. The core 3GPP model is a hierarchical model featuring management nodes or agents at each

level of the structure as illustrated in Fig. 4.2 taken from [107]. The network reference model contains three categories of node/agent – the Network Managers, Domain Managers, Element Managers and Network Elements forming the base layer of the structure. Each of which describe in general terms different stages in a communication network.

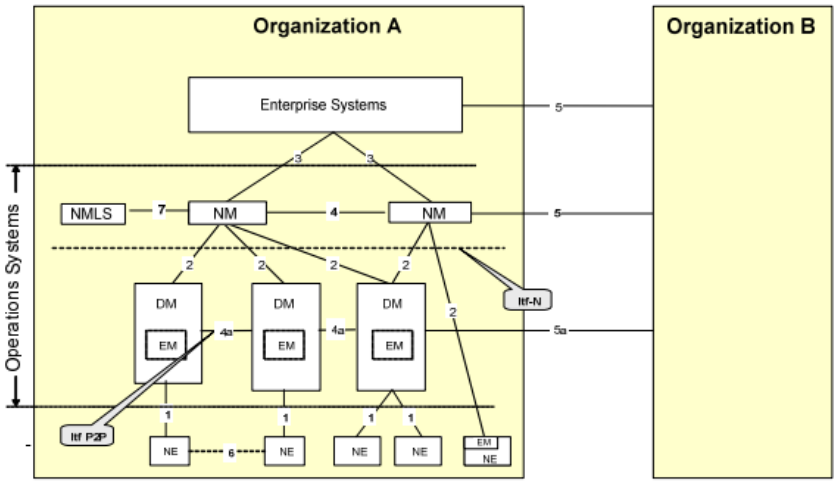


Fig. 4.2 – Management Reference Model

The authors of [106] use this structure to define four levels of Self-organisation management with respect to the placement of decision making. The approaches are as follows:

NM-Centralised Self-Organising Network (SON) – Performance data is supplied from the network elements (NE) to the Network Manager entity (NM), the NM then processes this information before determining is an architectural change is warranted.

DM-Centralised SON – Similar to the NM-Centralised approach with the exception that the sphere of influence of the SO controller is limited to the NE's under the jurisdiction of the relevant DM. Several DM entities may make restructuring decisions independently of neighbouring DM entities – as each one is responding to a set of local conditions.

Distributed SON – The network elements themselves make self-organising decisions based on reports from user equipment (UE). Updates from the UE devices are quicker and network elements can communicate with one another, however this approach is more localised and each decision does not have access to global information.

Hybrid SON – A combination of distributed and centralised approaches operating within the same scenario, combining the faster update times of the distributed version with the

wider scope of the centralised approach. Some elements of local reconfiguration are handled by distributed approaches while global decisions are handled centrally.

Each of the different levels of self-organisation relies on data retrieved from the communication network itself as a method of determining its current performance through continuous monitoring. A further implementation of the SEMAFOUR approach [109] to self-organisation and applying the 3GPP [110] standard involves the monitoring of Key Performance Indicators (KPIs). The KPIs monitor certain parameters such as network capacity, network coverage, call drop rate, handover success rate, or cell load. The KPIs are then used to define a set of targets which may result in a potential architectural change an example of such targets are presented in Fig. 4.3

- Dropped call rate < 2.5% (indicates the percentage of dropped voice calls due to, e.g., failed handovers or bad radio conditions)
- Cell load < 90% (indicates the used radio resources per cell or sector)
- Handover success rate> 99.5% (indicates the percentage of successful handovers between cells or sectors)
- Energy consumption < 80% (indicates the average consumed energy by the base station compared to the maximum energy consumption)

Fig. 4.3 – Example KPI Targets within a Mobile Communication Network

The use of KPIs reinforced the need for integrating continuous performance monitoring within a self-organising architecture, and the performance monitoring data is then transmitted to a location within the architecture responsible for performing the decision making. The KPIs formed the trigger factors for initiating an architectural transition event, when one or more of the performance metrics exceeded a designated threshold the KPI was considered to be violated and therefore indicating that a transition may be required. This research also indicated that there is scope for a hybrid self-organising architecture in terms of the smart grid implementation. Threshold comparisons and performance monitoring would be handled by the individual agents themselves whereas transition events would be initiated by an entity further up the hierarchy.

4.3.2 Sensor Networks

A second source of literature on self-organising architectures was found in sensor networks, due to the natural similarities between the field and the objective of developing self-organising smart grid architecture. Customer level agents and sensor entities within a sensor network exhibited similar properties because some of the core functions were shared

between the two research domains. Both architectures involved low level entities responsible for taking measurements, which were then passed to data collection nodes and eventually to a central data store. As with the research presented on communication networks, the entities involved in a sensor network were subject to different challenges as result of their environment, challenges which were not necessarily present in a smart grid environment. These included agent mobility, nodes with limited battery life and therefore greater agent turnover.

The first example is presented by the authors of [111], which focuses research on communication pathfinding between members of an underwater sensor network. The sensors aim to transmit information to a data collection point in the form of a sink node on the surface by passing data to neighbouring sensors. The authors describe a process wherein the sensor nodes send discovery – "REQ" – messages to nodes in its vicinity. The objective of neighbour discovery is to build a path from the sensor to the sink node on the surface, so only nodes closer to the surface respond to the "REQ" messages with a "RPLY" message. The source sensor node then assesses the time delay between the "REQ" and "RPLY" messages to determine which node is the best node to connect to. A stand-off period is imposed before selecting the desired neighbour to ensure that "RPLY" messages have been received. This communication sequence is documented in Fig. 4.4. Each sensor aims to connect to the neighbour with the shortest response time, and one which will remain in communication range for the longest time before being swept out of range by underwater currents.

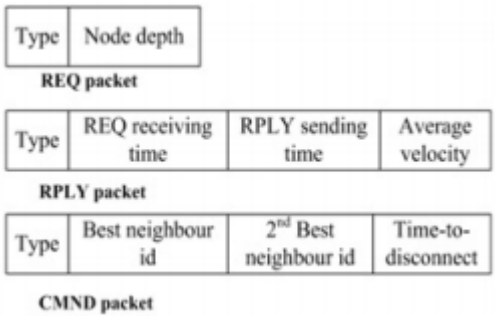


Fig. 4.4 – Neighbour Discover Packet Format

The process is similar to the connection finding method as applied in [105] as each sensor ranks the connections before selecting which node to form a link with. An alternative approach to sensor organisation is by clustering sensor nodes into partitions centred on an effective aggregation node. The authors of [112] suggest a method for structuring the sensor

network in response to placement of mobile cluster-nodes for the purposes of data collection. In contrast to the research presented in [111] this process is primarily centrally driven. An initial phase determines the number of clusters within the area covered by the sensor network; these clusters are then sub-divided such that the delay between the nodes within the cluster and the mobile actor data collection node is below a certain threshold.

These examples indicated that the need for an initial configuration stage is important for the development of a self-organising architecture, to allow the customer agents to form their own connections with the aggregation layer. Several implementations employed a ranking system to evaluate which data collection nodes would be the most applicable depending on properties endemic to the environment the nodes are active in.

A configuration stage would be supported by the ability to put nodes partially to sleep if certain functions are not needed by the authors of [113] regarding a set of mobile sensors in a wide area network. The sensors have two sets of internal behaviours each operated by an independent power supply. The first behaviour handles navigation and GPS communication whereas the secondary behaviour deals with inter-sensor communication. When the GPS elements are not required, that aspect of the system enters a dormant state while the listening behaviour remains active. Therefore if an aggregate did not receive any connection requests it would enter a dormant state, but any communication and listening behaviours would remain active in the event that it was to receive a signal to become an active participant in the architecture.

A repeated Self-organising solution within general sensor networks is Energy-efficient aDaptive hiErarchical and robusT Architecture (EDETA) which is the subject of several papers discussing self-organisation in the context of sensor networks, is initially proposed in [114] and applied in [115] and [116]. The EDETA approach describes a two part initialisation stage before the sensor network enters an operational stage. As per the original definition in [114], part one of the initialisation stage involves the election of a data collection node which acts as a cluster head node for a group of connected sensors. This process assumes a population of identical components, and within that population certain nodes are promoted to form a higher tier of sensors in the hierarchy whilst still performing core sensing responsibilities. Part two of the initialisation stage is to connect sensor nodes to the elected cluster-head nodes, a cluster-head has a limited capacity for incoming connections – therefore will reject requests once that capacity is reached. There are two

limits on the number of connections, a soft limit defines the number of sensors the cluster-head node will allow connection requests from, but the node will still accept connections if the request is for a 'last resort' connection. However the hard limit represents the absolute capacity of the cluster-head and no requests will be accepted if this limit has been reached.

Once the initialisation process has been completed, another function of the EDETA approach is the provision of substitute agents, which would replace a cluster-head node in the event of failure. The substitute continually monitors the connectivity of the cluster-head node so that it can detect if the node has failed, and if a failure is detected it assumes the role of the cluster head and informs the other sensors of the role change. This action aims to ensure that data is not lost and communication is not interrupted in the event of node failure.

The processes outlined in EDETA are very similar to those used in HARP [117], which promotes self-organisation through two phases – a set-up phase and a steady state phase. In HARP's initialisation phase, base sensor nodes transmit "JOINRQ" messages to the previously selected cluster head nodes. A JOINRQ message contains information about the sensor node including remaining energy and transmission power. After the "JOINREQ messages have been sent, the cluster head nodes broadcast "ASSOCCH" messages which contain a list nodes which have been approved connection. If a sensor node receives multiple ASSOCCH messages – it communicates to the cluster head with the lowest communication delay. The HARP approach also includes provision for a substitute node, to allow a cluster head to be immediately replaced in the event of failure. As in EDETA, the substitute node is pre-selected during the initialisation phase of the process – once selected the ID of the substitute node is transmitted to the sensor population. To take advantage of this process the self-organising architecture would require an initialisation phase, combining the communication structure of the EDETA approach supported by the connection ranking processes outlined earlier under mobile communication approaches.

One of the proposed applications of the EDETA solution is in fire detection as presented in [115], whereby 30 smoke detectors were placed over an ever increasing target area – to test coverage capabilities. However the authors of [115] don't pay too much attention of how the protocol interacts with the set of sensors – outlining the structure of the protocol as copied from the overview publication of [114] followed by a discussion on available sensor technologies. A later paper featuring an underwater sensor network proved to be more

illustrative of the practical applications of the EDETA method [116]. The underwater sensor network operated with two population scales of 100 and 200 sensor nodes and three coverage area scales, with a single centrals sink node, performing the role of the observer agent. The communication interval between updates from the sensor nodes to the cluster-head is every 250s – which is notably longer than the interval presently used in the current MAS simulation and several smart-meter projects.

From this investigation it was clear that an initialisation stage and that such a stage would be driven by the ability of the customer layer to rank potential connections to members of the aggregation tier. Furthermore aggregation agents were to play a more involved role in the process than initially outlined in the examples derived from communication network literature; these agents would have the ability to reject incoming connection requests based on connection capacity. Additionally the aggregation agents would have the responsibility of selecting a substitute agent which was a customer agent promoted to take on aggregation duties in the event of failure.

4.3.3 Vehicle Networks

A further research domain where self-organising properties and techniques are present was within smart-vehicle and transportation networks. Like wireless communication structures, one of the core driving factors is agent mobility. A road network would be under the supervision of a coordinating controller or management system and occupy a static geographic area but the number of entries operating within that network is subject to continuous change. Therefore research in this area was considered

The first example examined traffic management from a higher-level perspective as an implied multi-agent system detects and analyses congestion across the road network as discussed by the authors of [118]. The research outlines a method for allowing smart-vehicles alternative routes based on traffic congestion data, in contrast to the previous examples the network management is primarily delivered through simulation rather than interaction between physical entities. A control centre maintains a model of the road network and the vehicles using it, the network itself is dissected into road segments – referring to unbroken stretches of road between junctions. Smart-vehicles communicate their position to the control centre and receive a representation of the model so each vehicle is aware of the decomposition of road segments.

The objective of the system is to calculate how long a vehicle spends travelling along each road segment – much in the same way that message transmission times are calculated. The vehicle announces to the nearest server when it has entered a segment, and when it has exited it - along with a unique vehicle identifier to distinguish between vehicles entering and exiting the area. These times are logged and compared to a pre-calculated time representing clear passage through the area. The pre-calculated time is a function of the length of the road segment, its speed limit and number of traffic signals along its length. If none of the recorded transit times exceed the pre-calculated estimate, then the road is declared clear. Otherwise if transit times are recorded above the threshold, it becomes indicative of potential congestion within the road segment. In the paper the local-server in charge of the road segment extracts the two/three longest transit times as a measure of network congestion rather than an overall average.

Upon calculation of the level of congestion on a given road segment, incoming vehicles need to be informed such that they can make organisational decisions about which road segment to take to complete the journey. The authors recognise that decisions can be influenced by human behaviour depending on mood, journey purpose, visual information or local knowledge – each of which would influence their decision in response of recommendations delivered by the monitoring system. Two potential choices are given to the vehicles subscribing to the management system:

Stay on track Strategy - Under this strategy, vehicles will continue to travel along the congested route up until the delay caused by the congestion issue exceeds the length of the offered alternative route. This operates under the logic that changing route to avoid congestion may not be the best solution for the vehicle. Therefore until such time that the alternative route actually offers a quicker journey time – avoid using it.

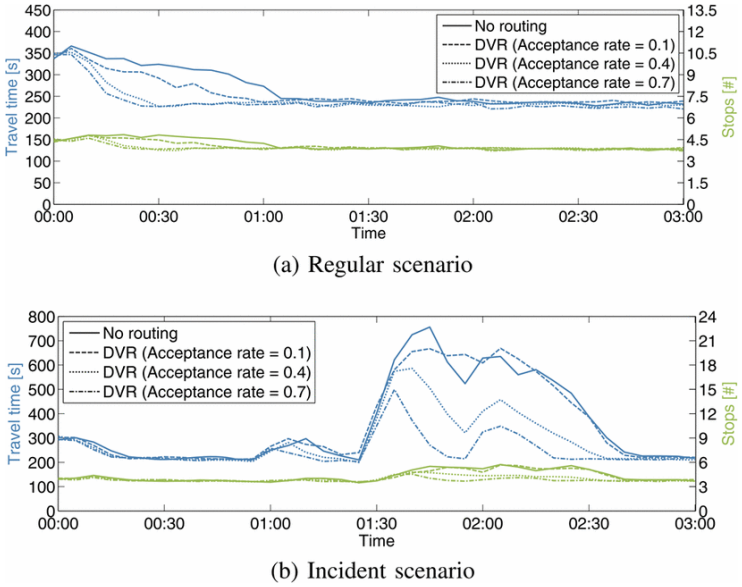
Immediate Evasion Strategy – this approach involves the vehicle opting for the non-congested route without considering the time-delays involved in the detour.

In some respects the first option could be considered customer centric –where the needs of the customer are put first, while the other is network centric as further congestion on the initial route is limited through re-routing traffic before it becomes a more significant problem.

This research centred on the use of performance monitoring data used to inform reconfiguration actions, but also demonstrated that a reconfiguration action had the potential to be over-ridden by expert knowledge. Furthermore it also discussed that the act of taking no action in response to performance metrics exceeding thresholds is a valid course of action in the event that the transition event itself would prove to be more disruptive than the current congestion issue. Therefore it was important to consider if it was ultimate necessary to perform a transition in the event that even if thresholds have been exceeded, the effective impact on the network is relatively small. An immediate evasion strategy would involve the self-organising architecture performing a transition event for each metric threshold violation.

An alternative traffic management and congestion avoidance mechanism is outlined in [119], the solution uses an approach similar to the 'stay on track strategy' implemented in the previous example. The difference being that the vehicles themselves are not necessarily involved in the process, potential expansions suggest the introduction of car-to-infrastructure communication. Instead the junctions within the road network form the base layer of agents, and the vehicles are more analogous to data packets within the network. Therefore the scenario becomes a load balancing problem – minimising the amount of cars trapped at each of the junctions by routing the vehicles more efficiently. Each junction has an observer/controller agent which assesses the average waiting time for vehicles due to red light conditions – which can be compared to a message queue assessment. The junction then informs drivers of the least congested route via display boards if it knows that the suggested route would offer a time saving. To make these assessments there is no central server as proposed in [118], instead junction controllers communicate with each other, in what is effectively a flat architecture such that the junctions can update their internal routing tables with an estimated journey time between nodes – an estimation which is composed of road speed limits and waiting times as collected by the observer/controller agents. Therefore in this case the provision of self-organisation is performed through redirecting traffic between nodes in the road network based on performance monitoring information.

The results of this implementation indicated that under normal conditions the improvement in traffic flow was limited, as illustrated in Fig. 4.5(a). However under a scenario when connections within the network were blocked, the system delivered more effective results as presented in Fig. 4.5(b).



(a) Regular scenario

(b) Incident scenario

Fig. 4.5 – Adaptive Traffic Control Results, with and without Incident

It was not able to eliminate congestion building due blocked roads, but it was able to prevent vehicles from attempting to use the severed connections. This enforces the use of self-organisation for the purposes of resilience – detecting a fault within the communication architecture and modifying the recipient addresses of the transmitting agents to prevent messages being transmitted to a failed node. The increase in overall congestion is expected in these circumstances, as the volume of vehicles (of communication messages) then has to be passed onto the remaining active nodes and ultimately increasing the communication load.

Further examples don't necessarily implement self-organisation from the perspective of congestion and traffic management – instead considering the vehicle network in the context of a mobile sensor network and improving data collection as presented in: [36], [120], [121]. However these solutions are predominantly aimed to counter the condition of node mobility, which depending on the road network under surveillance can involve nodes travelling at significant speeds, which is not a situation encountered in the smart grid monitoring system.

This research indicates that beyond an initialisation stage, the use of self-organisation was more effective when the system under observation encounters a condition outside of normal operating parameters. A failure of an aggregation agent would correspond with a blocked road and represent an event whereby triggering a reconfiguration event would deliver a performance improvement. A sudden rise in congestion would also be created if a cyber-

attack event created a stream of attack traffic aimed at interrupting the flow of legitimate data.

4.3.4 Multi-Agent Systems

Multi-agent systems was another key research domain in which self-organising properties are exhibited and therefore could be investigated for the purposes for integrating techniques into a self-organising architecture. Unlike a sensor or communication network a multi-agent system often required a greater degree of interaction between entities in terms of retrieving monitoring data, processing that information and responding with control actions. Therefore the control objectives of the system also had to be factored into the self-organisation process.

The first example considers reconfiguration on the basis of partial agent failure as documented by the authors of [122] whereby an agent community consisting of robots on an industrial production line is examined. The case study presents a scenario where one of the robots loses partial functionality, and thus the agent population is restructured in order to prevent the failure halting the production line. The authors suggest the idea of behavioural redundancy such that in the event of failures, the defective robot switches roles with another functional agent such that the defective agent operates in a position where the defects will not influence performance. For this process to work the agents need to be equipped with multiple behaviours for multiple roles within the network – allowing adapting to a new position within the community. To perform this role swap, the robot/agent needs to be aware of its own performance metrics and capabilities to decide whether or not it is able to continue performing the role it is currently allocated to. Once a defect has been detected, the affected agent contacts its neighbours to ask for a position change – explaining which failure has taken place, and what functionality has been affected. After making the help request, and agent which possesses the ability the defective agent has lost responds to the defective agent and the others in the system as an acknowledgement of being willing to help. Once the transition has taken place the defective agent concludes that it doesn't possess the complete behaviour set required for its new position in the system. Therefore triggering a second transitional phase – the sequence of events is represented in Fig. 4.6

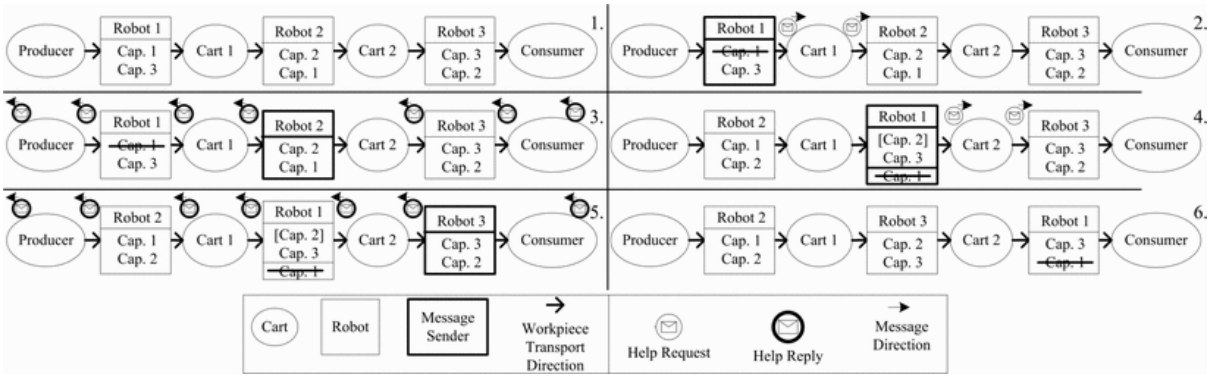


Fig. 4.6 – Role Swapping in a Self-Organising Multi-Agent System

This research demonstrated that building behaviours into agents associated with other roles within the architecture improves the flexibility of the system. Such that agents can be transplanted into alternative roles within the system in the event that either the original agent had failed or additional support is required. Therefore agents could be designed to accept differing responsibilities if they are promoted or demoted. For example under the system described in the EDETA strategy the substitute agents were identical to the cluster-head agents they were designed to replace, however it became evident that it would be more applicable to source substitutes from the a different agent group. This was because the customer layer had a larger accessible population, and therefore in the event of aggregate failure it was more appropriate to source a replacement from a tier with sufficient resources.

Additional work in self-organising multi-agent systems considered applications within the smart grid domain where the focus is directed towards self-healing as documented in the following papers: [123], [124], [125]. Those papers separate the self-healing concept from the self-organising whole and apply is specifically to the control problem while the MAS remains static throughout.

A further paper details a process for forming the core components of a self-healing process as described by the authors of [126]. These core components form a three layer approach which governs a self-healing process; these three layers are as follows.

4. Fault Detection Layer
5. Fault Diagnosis Layer
6. Corrective Action Layer

The corrective actions discussed by the authors are focussed towards changes made to the electrical network rather than the communication layer, but the core principals can be adapted to the management of agent interactions. The initial layer – fault identification –

takes place at the sensor level, which would equate to the customer agent level – assuming that these sensors which have some degree of awareness to notice that a particular variable is no longer within limits. Phase two is the diagnosis layer, which determines what has failed and if so propose a temporary solution if possible, while the final layer instigates the architectural change to prevent further damage to the system and restore service. This reinforces the idea that in the aim of developing a self-organising system it is important to monitor agent performance and variables which are not involved in the actual provision of control and general operation of the network.

Aside from the smart grid domain the concept of self-healing through changes in architecture has been considered for web services [127], [128], [129] whereby multi-agent systems are applied. In [127], a two stage process is employed in the event of a failure within the agent community. Before an action is conducted a diagnosis agent analyses faults, a fault is defined as an instance where a performance metric exceeds specified thresholds usually in the form of various timers. If a number of faults are detected over a period of time – this is deemed to be a failure and as such is referred to the repair agent. This repair agent has the choice of two courses or action – attempt to re-establish the connection between customer and supplier or find a replacement agent which can carry out the tasks the customer is requesting. The replacement concept is a reflection of the substitute node concept presented in [114]. Except that in the case of the web-service scenario not all agents will be involved at any given time, so an inactive but functional agent can be used to substitute for the agent who has triggered the fault condition. The performance monitoring element of the process introduces a series of performance metrics measuring communication and agent effectiveness through a series of timers. These metrics are presented in greater depth in [129], illustrated in Fig. 4.7.

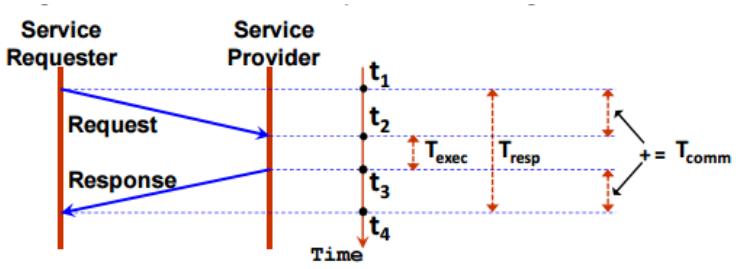


Fig. 4.7 – Communication Performance metrics

The figure presents several differing time intervals which provide five sets of performance measurement these times are described as follows:

- **The Response Time**: defined as the time elapsed between sending a request and receiving its response; Tresp = t4 − t1
- **The Execution Time**: defined as the time elapsed for processing a request; Texec = t3 − t2,
- **The Communication Time**: defined as the round trip time of a request and its response; Tcomm = Tresp − Texec
- **The Throughput**: defined as the amount of requests that can be processed in a specified period of time; Throughput = Number of requests/period of time,
- **The Accuracy**: defined as the success rate produced by the service; Accuracy = Number of successful responses/Total number of requests.

Each of the parameters would be associated with a threshold governing the anticipated communication performance between agent pairs. If these measurements increased significantly this could be an indication of other events taking place within the architecture, either in the form of poor local node performance or the influence of a cyber-attack.

4.3.5 Smart Grids

While other research domains have alluded to applications related to smart grids, such as the smart-meter communication networks presented in [105] and self-healing multi-agent systems. It was also important to consider research conducted within the power systems domain because while the exploration of self-organisation in power systems is not presently extensive the development of advanced and increasingly intelligent monitoring and control methodologies increases its applicability. This is driven by the need for decentralised control and increasing quantities of sensors and monitoring technologies applied at the LV end of the network. Due to the high number of potential control scenarios involved with smart grid management the idea of self-organisation can be applied to a considerable variety of contexts.

On example is concerned with organising charging priorities for electric vehicles with the aim of reducing the impact on the network through staggering charging patterns [130]. Because of the introduction of a control element the requirements of a self-organising system become notably more diverse, moving beyond topological changes to the architecture. The authors of [130] integrate the concept of self-organisation through

interactions on the lowest layer of the smart grid architecture, whereby the needs of the connected EVs dictate the control priorities. It could be argued that the self-organised EV charging concept is more of a decentralised control approach as no network configuration or control changes are made to the network – only the order and duration of which an EV is charged.

A second example examines the smart grid as a cluster of interacting sensors within the context of voltage quality monitoring, in [131], the authors present a system wherein the sensors behave both like network nodes in a communication network or sensor grid and like agents in multi-agent system. The core component of the self-organisation is delivered through the communication network and routing paths between measurement and data storage. The roles and number of data aggregation points is not varied within the proposed solution, instead variations in the monitoring process are predominantly aimed at calculating a voltage quality index within the cluster of sensors observing a section of the electrical network. An architecture is proposed for the purposes of voltage monitoring which is based around node coupling is composed from the connection matrix of the underlying electrical network topology, again with the primary objective of determining voltage quality. Specific consideration with respect to drivers for variations in the architecture or decisions that may influence a topological change is not present within the research.

A third example of self-organisation within smart grids illustrates the benefits of implementing the hierarchical structure using agents rather than a conventional client-server architecture [132]. A specific focus is placed on reliability and robustness as a predominant driver for implementing self-organised monitoring systems. The authors introduce agent based technologies and an alternative to the strictly centralised traditional monitoring approach in system monitoring, two agent classifications are involved Intelligent Distribution Agent (IDA) and Power Quality Agent (PQA). These to agents form a hierarchical structure where the PQAs form the base layer, performing the monitoring responsibilities and processing the collected data. Processed results are then communicated to the IDA, acting as a local controller for the area served by the collection of PQAs. It analyses the incoming updates from the PQA population and compares the results with a series of thresholds. Each PQA can only communicate with its immediate neighbours and the IDA – thus lowering the bandwidth requirements to accommodate the system. Unlike a mobile communication network or an un-tethered underwater sensor network – the nodes

forming the majority of the population are static, positioned according to measurement availability or network points containing sensitive equipment. But self-organisational concepts are applied through modifying the virtual topology rather than the physical one – through changing agent responsibilities and communication routes. In this instance the driver for re-configuration is agent failure, with a view of improving robustness. As the architecture diagram in Fig. 4.8 illustrates, each monitoring cluster contains a single, central IDA – which is a single point of failure thus making the agent community just as fragile as the centralised client-server paradigm is it intended to replace.

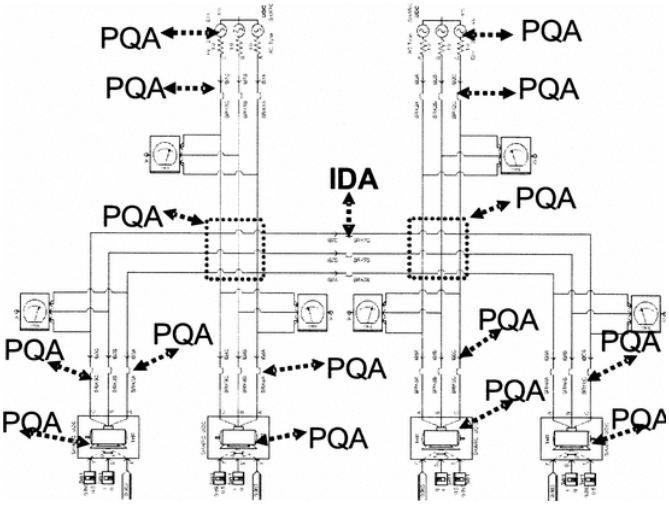


Fig. 4.8 – MAS Topology for Power Quality Monitoring

Therefore in the event of failure the authors suggest that it would be replaced by a PQA, promoted to assume the responsibilities of an IDA as illustrated in Fig. 4.8. The process for selecting promoting one of the existent PQAs involves ranking the set of potential PQAs available for promotion with respect to a series of criteria – based upon expert assessments.

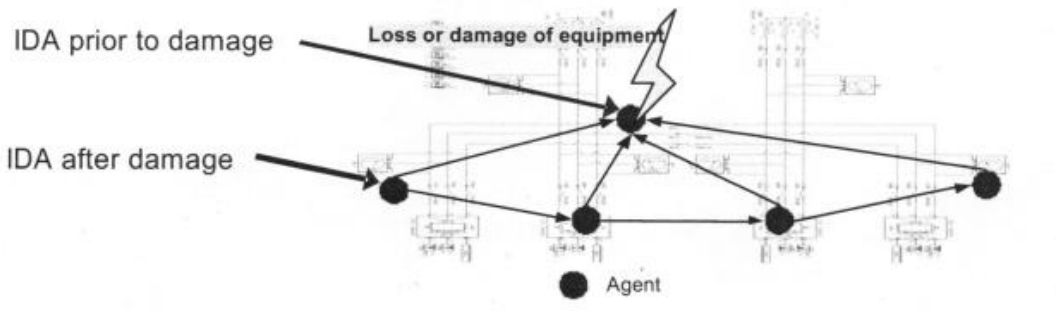


Fig. 4.9 – IDA Promotion - Post Agent Failure

The key criteria involved are: Computational ability, Computational burden, Adjacency to other PQAs and Adjacency to other IDAs; therefore assessing the nearby PQAs for their ability to host the IDA behaviours and their location within the network.

This research furthers the concept of using substitution as a method of replacing failed agents within a system, and that those substitutes do not necessarily have to be of the same agent type as the agent that has failed. As a result additional behaviours need to be embedded within the agent population such that they can assume new responsibilities. The research also indicates a system of determining which agent should be selected as a substitute in the event of failure, the EDETA approach pre-selects substitutes but a selection process could still be applied such that the pre-selected agent is the most accessible replacement when it is needed.

Another driver for self-organisation within the smart grid domain is the influence of potential cyber-attacks compromising certain sectors of the network. The authors of [133] present the concept of controller switching in the event of a communication jamming attack affecting the connectivity between smart-meters and the local controller node. In a similar concept to the previous example the non-mobility of the smart-meter nodes means that the re-configuration is conducted by re-routing communications away from the affected area. The process involves using channel hopping to counteract the influence of the jamming signal; at this point a self-organised approach isn't in play as the communication remains between the same pair of nodes, the customer and the local controller. However if the solution cannot avert the issues created by the jamming signal then the smart-meter needs to look elsewhere –and therefore becomes the architect of the self-organisation process. The system then retains the channel hopping approach, but widens the range by including the open channels on nearby controllers. To do this the smart-meter node contacts the neighbouring controllers and requests it's channels hopping sequence – which defines which channels are in use for a given timeslot. From this information the smart-meter builds a controller switching matrix which defines which controller will be contacted on each timeslot, based on channel availability from the sequence data. While channel hopping processes are not simulated in the present model, as it focusses on the beginning and end points involved in a communication pairing rather than the communication technology itself, this example is another application of self-organisation for the purposes of self-healing and robustness.

Self-configuration can also take place at the transmission level where the voltage levels are much higher. In this case the smallest nodes are the sensor nodes instead of customers which communicate with regional aggregators. The authors of [134] propose a self-configuring architecture focussed on connecting transmission line sensors with gateway

nodes which connect the sensor network with the remainder of the communication infrastructure. However this approach is presented as a solution to automated gateway detection for new sensor installations, while it is important to develop approaches for new nodes to integrate with the system the paper contains no indication of repeated architectural transitions and triggers for more wide spread action.

4.4 RESEARCH GAPS AND OPPORTUNITIES

The majority of the papers presented self-organisation based solutions to specific problems within the target domain and these solutions were primarily centred on either load balancing or recovery from failure. For example, the authors of [132] presented a self-organising network for the purposes of recovering from a fault at one of the nodes – responding through re-allocating the responsibilities of the failed node to an active alternative. In this example the concept of self-organisation is solely applied to resolving the issue of agent failure, whereas [105] documents an approach for allocating new nodes a connection to an aggregation point in a system with a dynamic population.

In [114] an initialisation phase was described and in [109] a method for deciding how to modify the network is presented. In a smart grid environment under the control of a self-organised MAS, a more complete approach was required, one that exhibited several of the aforementioned qualities and techniques, plus others from across the self-organising community. This variety of solutions is a product of the widespread applicability of self-organisation and the multitude of potential interpretations and implementations, but even after narrowing the subject domain to networks and architectures – different research streams have their own specificities.

One limitation of the present set of research examined is that there appears to be little consideration for the time-scales involved in any of the phases of a self-organised system. In terms of determining how long it takes to perform customer-aggregate allocations for various customer populations. Whether differing techniques would achieve convergence faster, or achieve a more even distribution of the number of connections per aggregate. The consideration for differing time scales could also be applied to responding to node failure, replacing a failed node is not an instant process and therefore there will be consequences for an elongated transition time. Furthermore the documented literature does consider the

implications for control actions during the node replacement stage and whether some control objectives would be more severely affected than others.

Different solutions also focus on different performance metrics for ranking the potential connection options to the data collection points. The authors of [111] use a combination of communication delay, and predicted availability. Because nodes in the smart grid context are of a fixed position the issue of agent mobility affecting availability is removed, thus leaving decision making to be based on communication delay. On the other hand [132] uses physical proximity as one of the comparison criteria alongside computational burden and available resources to determine the most applicable agent. The latter may incur a larger communication overhead to transmit the additional metrics but prove more accurate in terms of fairly distributing connections.

One notable omission from the set of sources referenced is an investigation into the overhead produced by the additional monitoring and diagnostic messages required in the context of a self-organised system on top of the base-level communication for a given scenario. For example, hard-coded MAS will have all of the connections pre-loaded and therefore wouldn't need to undergo an initialisation phase which involves the transmission of various discovery messages. In order to make rational decisions about performing architectural changes, knowledge of the current system state needs to be collected and then transmitted to an agent with the powers to act upon it if necessary. This will ultimately increase the volume of data that the agents will have to process, and it raises the question: would this additional communication load and processing demand interfere with control processes.

In terms of opportunities, there is certainly a case for implementing self-organisation within the context of smart grid management. To achieve self-organised status the agents need to be able to perform network discovery themselves, make their own connections – a concept which is noted in communication architectures and referenced with respect to smart-meters [105] and sensor networks [114]. The case for self-organisation due to agent mobility isn't necessarily applicable to the smart grid domain, as the agents in this case are of a fixed location – domestic properties, generators etc. which accounts for the limited research output on self-organising communication/control architectures for smart grids. Yet the emergence of electric vehicle ownership would present a differing self-organisational challenge.

However a strong case is present for self-organisation in response to failure – whereby the agent network has to replace a failed agent in order to maintain a chain of communication or replace a lost controller. Failures could also reduce the maximum incoming or outgoing data capacity from a particular agent, therefore forcing the network to modify the distribution of communication load by either using up capacity on neighbouring agents, or introducing formerly dormant agents into the equation to take on the additional load. This scenario invokes the second rationale for a self-organising approach, which involves load balancing specifically of the communication load – identifying aggregates with higher than average congestion or reactivity times and instructing them to lay off connections to neighbours if possible. Self-organisation is relevant to the current research and in terms of managing the control and communication architectures in a smart grid concept, but the discrete transitions from the Disaggregated to Base to Cluster to Tiered communication structures may not be as applicable as greater flexibility would be beneficial.

A further trigger for self-organisation is the presence of a cyber-attack event, whereby members of the agent population are compromised and therefore experience a reduction in functionality. The presence of cyber threats is an emerging topic within the smart grid and power systems domain and therefore requires as much attention and consideration as hardware or software failure related triggers. Failures within the network could easily be the result of an attack event aiming to damage control functions or compromise security measures.

Also it could be considered an opportunity to amalgamate several different approaches to individual objectives into the same system – from an initialisation phase, conventional operation, performance monitoring, response to failure, load balancing, and architectural change decision making – all with respect to ensuring that control performance goals are maintained. Several techniques and approaches can be adapted and integrated to form a more complete self-organising architecture.

4.5 DISCUSSION AND CONCLUSION

Given that the current set of research examined in this chapter was spread across several subject areas, tackling several problems there was no single standout implementation which could then be translated into the smart grid domain. Furthermore it was important that in the objective of developing a self-organised MAS system the properties of the system

adhered to the properties as outlined in definitions provided by conceptual papers [37] [38] [102]. The system benefited from adapting multiple approaches from across the literature to solve the differing problems involved. Overall there was a reasonable degree of applicability for the principles and concepts of self-organisation within smart grid solutions. The most applicable of these was the response to failure either due to a technical malfunction or under the influence of a cyber-threat. A failure of a controller or data collection agent would interrupt the observability or controllability of the smart grid network, and therefore placing a great deal of importance in fault recovery with view towards continuous operation. The following papers [130], [131] and [132], illustrated that existing research considers the importance and relevance of self-organisation within the smart grid domain, and the examples cite differing aspects of smart grid applications which would benefit from these techniques. In [132], the authors use self-organisation for the purpose of recovering from agent failure through installing redundant behaviours and agent promotion. The other examples focus on a version of the load-balancing problem whereby EV charging patterns are organised to reduce impact on the grid and ensure that vehicles are still charged. These don't take into consideration the architecture of a given system, or make topological changes to the structure of the network – but support the idea of the agents in the lowest layer of the hierarchy taking a leading role in determining which agents/nodes to connect with.

As per the processes outlined in, [114] [115] [116] regarding the EDETA method – a self-organised structure needed to go through an initialisation phase at start-up. This phase involved customers selecting their own aggregation agent to make a connection to; these connections were selected on the basis of agent availability and capacity. The rationale for including this phase is with respect to the characteristics presented in [38], whereby the authors indicate a property entitled *"Endogenous global order"* which essentially suggests that the set of agents involved within a self-organised system are required settle into a stable operating state. This entailed customers finding a connection and thus establishing a link along which demand data and control alerts can be transmitted. In the context of the EDETA sensor network concept, the connections are formed based on proximity to the node requesting a connection and whether or not the target node has enough capacity to accept another connection. When establishing data and control connections between customers it was important that the customer was able to build a set of information such that it could make an informed decision as to which aggregate/controller to communicate

with. This required a series of messages similar to the REQ/RPLY messages used as part of the neighbour discovery process described in [111], where the difference between a transmitted REQ message and a received RPLY message is used to measure the communication time between sender and target. Channels with quicker response times then represent a more applicable connection.

Outside the initial connection phase, examples in the literature both application specific and general principle pointed to a necessity for continual measurement which in turn reflects the objective of future power systems to maintain an increased level of observability. The authors of [102] introduce an observer element of which the sole responsibility is to retrieve performance metrics from the set of agents operating in the system. This is a centralised approach, where a single element is in charge of providing a controller with information on the system state, the controller then decides what actions need to be taken. An alternative approach to performance monitoring, with respect to agent failure detection is presented in [127], which follows a similar approach to that introduced in [102] – instead an observer, a diagnosis agent examines the performance data to look for instances where measured parameters exceed the designated limits. As the goal of the system is detect failure or sub-optimal performance the parameters are centred on message response times. The diagnosis agent flags each instance of an overly long response time as evidence of a potential failure; multiple instances are then flagged up to the next agent as a call for repair or replacement. A third example of performance monitoring is taken from [118], where updates are again published to a central server to be processed. However in the interests of improving scalability, it may be more appropriate to disseminate the monitoring process at a more local level.

Each agent being able to assess its own communication times, congestion and control performance where applicable avoided the necessity for large numbers of messages in a highly populated system being transmitted to a central agent. There was still a case for the existence of a specific agent responsible for handling architectural changes – as in [127] – an architect agent would not receive all performance data, but would need to establish a view of key agents and their connections. As individual agents detect an anomaly in the performance metrics – alerts were sent to the architect agent, to limit the amount of traffic that the architect needed to process. Instead of creating a separate controller entity as in [102], the architect analysed the set of alert messages to determine a common link. For example this could mean that several customers noted slower response times to the same

aggregate, therefore corrective action is then applied to the aggregate in question. There is the concern that the overhead presented by diagnostic messages – especially in relation to failure detection will contribute a number of messages to the total data traffic. Some of the metrics were embedded within agent messages, through adding fields to the message content: time sent, time received, processing time, congestion data etc. As managing data collection is recognised as a source of self-organising interest as noted in [36], [120], [121], [112]

As previously noted two objectives are common within self-organising literature – response to failure and load balancing. Because a smart grid system is highly dependent on the IT infrastructure and the communication links between components in that system, the load balancing issue would influence the handling of agent messages and control signals. A system with concentrations of message congestion is likely then to experience issues in terms of disseminating control signals and receiving the appropriate acknowledgement. Therefore it was important to include a process for monitoring communication load across the key bottleneck areas within the MAS. Some of this was handled during the initialisation phase through distributing connections, but as the system operates – agents may experience changes in communication intensity and therefore may request some of the load be re-allocated elsewhere.

The second objective is responding to failure, in [132] agents effectively bid to replace a failed agent – while in [114] key agents are assigned a substitute which is invoked at the point of failure. For the prospective MAS, a fusion of these techniques was more applicable to avoid delays when performing a bidding process post-failure. From the perspective of an aggregation agent it needed to select the most appropriate substitute – data extracted from the sequence of messages involved in making a customer connection was used as bid information. The customer agent with the lowest bid – in the form of round-trip message delay – was then selected as the substitute. This process is conducted during the initialisation phase, at such point that the aggregate had either reached maximum connection capacity or hadn't received a connection request within a timeout period. This pre-selected the replacement in the event of failure, the ID of which is passed to the architect agent for the purposes of knowing where to re-route communications post-failure. The architect also needed to know which agents had made connections with the failed aggregate, such that they could be contacted and informed of the identity of the substitute.

Finally the decision making process for selecting which architecture to transition into is a key aspect of a self-organising system, and in the context of the smart grid domain, this decision would need result in a stable configuration. It is not feasible to introduce several trial and error based stages before settling on a the next architecture structure as enduring multiple transition phases in a short period of time may lead to incomplete transitions and interfere with underlying control processes. A series of rules is suggested in [109] and [135], based on performance indicators. Combinations of performance indicators are mapped to actions or reconfiguration options – translating the concept into current project in conjunction with the performance monitoring flags can be used to form a table of actions based on events.

For example, if several customers contacted the architect with an alert message stating that they have received no contact from an aggregate agent – a rule will map that condition to the action of invoking the substitute to replace the failed agent. A second example might entail multiple messages from aggregation agents indicating that they are receiving connection requests after reaching maximum capacity. This would be mapped to an instruction to increase the aggregation capacity, to accommodate the additional customers. While some of the self-organising functions are completed in a decentralised manner through distribution across the overall agent population – the rule mappings would be hosted by the architect agent. This is reflective of the network management approach presented in [135] – where a network objective manager hosts the responsibility comparing performance indicators and their respective actions.

Overall there is no single approach that could be immediately translated into the context of the self-organising architecture documented in this thesis. In accordance with the conceptual research presented in [37], [38], [102] the system needed to adopt more functionality in addition to transitioning from one configuration to the next. Therefore an initialisation phase composed of elements from [111], [114] was introduced to establish connections between customers and aggregates/controllers based on communication delays and aggregator/controller capacity. During this phase agent substitutes are pre-selected. After the completion of an initialisation stage performance monitoring techniques were used to maintain observability. These techniques were based on the concepts presented in [105], [118], [127] and [129] with the exception that only instances of performance indicators exceeding threshold values are communicated to the architect agent. Finally in response to the alerts a hybrid approach to self-organised management will be introduced

as per [107] where different actions are completed by different tiers within the overall MAS architecture. Some load balancing actions can be handled in a decentralised manner through connection handovers from one aggregate to the next. Wider architectural changes, including transitioning from one core design to another through activating dormant aggregates or promoting/demoting agents will be handled by the architect agent. This agent should maintain a list of conditions and actions similar to those implemented in [109] and [135]. These actions could involve controlling whether an agent is dormant or not for the purposes of balancing communication load [106] or replacing a failed agent [132], [114]. For some changes, a 'stay on track' strategy [118] can be employed – mostly in the case of connection transfers – to check if transferring the connection would result in a performance improvement, to avoid unnecessary modifications to the system.

Overall considering the set of existing literature and the self-organising architecture required for the set of agents performing voltage control and data collection duties a three stage process shall be introduced. Stage one of the process consists of an initialisation phase where the agents can locate a set of potential aggregates and request a connection. Connections will be prioritised on the response time of the aggregate, and whether the aggregate has reached capacity. Once the architecture is fully connected the second stage begins, this stage consists of a conventional operation state and is focussed around performance monitoring, and the communication metrics retained by the individual agents will be given appropriate thresholds. Violation of those thresholds would constitute an error report.

4.6 SUMMARY

This chapter presented a range of different research topics in which concepts pertaining to self-organised systems were presented, in addition to exploring application specific techniques and approaches, underlying requirements were also examined. In addition to interrogating relevant literature in domain of self-organising systems the chapter also explored the gaps in the documented research. From these gaps suggestions for strategies which would then implemented in the development of a self-organising agent architecture were inferred. Finally conclusions were drawn with regards to elements of the reviewed literature which could be adapted and integrated into the developed architecture.

Chapter 5: Developing a Self-Organising Architecture

5.1 SYSTEM OUTLINE

Research into the concepts and mechanisms involved with self-organisation conducted in the previous chapter outlined that thee core stages were required to develop a system which could be considered self-organising. The processes involved needed to be adapted for implementation within the context of a smart grid architecture solution.

The first stage of operation was focussed on the connection of customer layer agents to the aggregation tier for the purposes of data transmission; this stage also included connecting customers to controllers. The second stage accounted for the majority of the runtime, where the agent population conducted performance monitoring, comparing individual variables against a set of thresholds. This performance monitoring stage was triggered once initialisation was completed, after each transition event the architecture reverted to the performance monitoring stage. Finally the third stage was responsible for the process of performing architecture transitions on the back of a decision making engine, and determined which of the available transition mechanisms needed to be invoked based on the data recovered during performance monitoring.

This chapter will discuss the development of each of the different stages involved within the self-organising architecture covering the communication requirements and the processes necessary for the stages. The chapter will examine the architecture transition functions and method for triggering and simulating attack mechanisms within the architecture.

5.2 INITIALISATION STAGE

The initialisation stage defined the initial start-up procedure in terms of how the agents form a communication and control architecture through contacting controllers and data collection points in the form of aggregation agents. The initialisation stage was adapted from mechanisms employed by the EDETA [114], HARP [117] approaches for sensor networks in addition to the ‘Tic tac toe’ [111] architecture also used for sensor communications.

The EDETA solution assumed a set of homogenous nodes and aimed to build a hierarchical structure for data transmission. Of this population a proportion were elected as ‘cluster head’ nodes and therefore formed the upper tier of the hierarchy. The EDETA and HARP approaches presented an initialisation stage whereby the nodes which were not selected to

be elected into cluster-head positions seek out cluster-head nodes for the purposes of forming a connection. The Tic-tac-toe-arch also presents an initialisation stage and introduces a communication protocol for selecting the relevant data collection point and for sending connection requests. Rather than using proximity, the approach uses the interaction between a sensor node and a cluster head node as a metric for selecting the most appropriate option. This interaction involves the base tier sensor nodes, sending REQ messages and receiving RPLY messages from the sink nodes. A similar approach has been applied in the current implementation through 'DISCOVER' and 'HELLO' message pairs.

An additional component of the initialisation stage is the selection of substitution agents for the aggregation layer agents. Substitutes are proposed as part of the EDETA approach. The cluster heads are allocated a substitute node in the event of failure, which may be a result of power loss in the context of the sensor networks involved. Therefore a substitute is selected to replace the missing cluster head and maintain the data link from the sensors to the surface node. This adds a degree of fault tolerance to the network.

Overall the present initialisation stage has been put into place on the influence of the EDETA/HARP approaches which present a multi-stage method of applying a self-organising architecture to sensor networks. Secondly the internal methodology for pairing customer/generation agents with an aggregator was informed by the Tic-Tac-Toe-Arch method. The time delay between the transmission of a 'DISCOVER' message and the reception of a 'HELLO' was used as a metric for selecting aggregate agents to connect to, this metric bears a similarity to the process applied by [105] when addressing the issue of smart-meter networks.

5.2.1 Establishing a control and communication connection

The first part of the initialisation phase was to establish a connection to a data collection point within the architecture, and to select a controller. A communication connection focussed on building a route from the customer to the observer so that demand information could be extracted and filtered to the top of the hierarchy. Generation agents are seen as customers from the perspective of searching for a communication connection, as transmitting generation output data was considered to be identical from a messaging viewpoint as transmitting demand. Whereas a control connection determined which tier of the hierarchy a customer will transmit control requests in the event of a voltage deviation.

The following diagram in Fig. 5. illustrates the initial negotiation process between a customer agent and an aggregate, each communication interaction was stored as a series of variables recorded as a ConnectionObject. A ConnectionObjeect was a java class written to manage the initialisation process from the perspective of the customer/generator agent – determining whether or not the relevant discovery messages had been sent and responded to. The class also calculated the response time, it is this response time that was used by the agent to decide which of the aggregates active on the network it wished to connect to as its first priority.

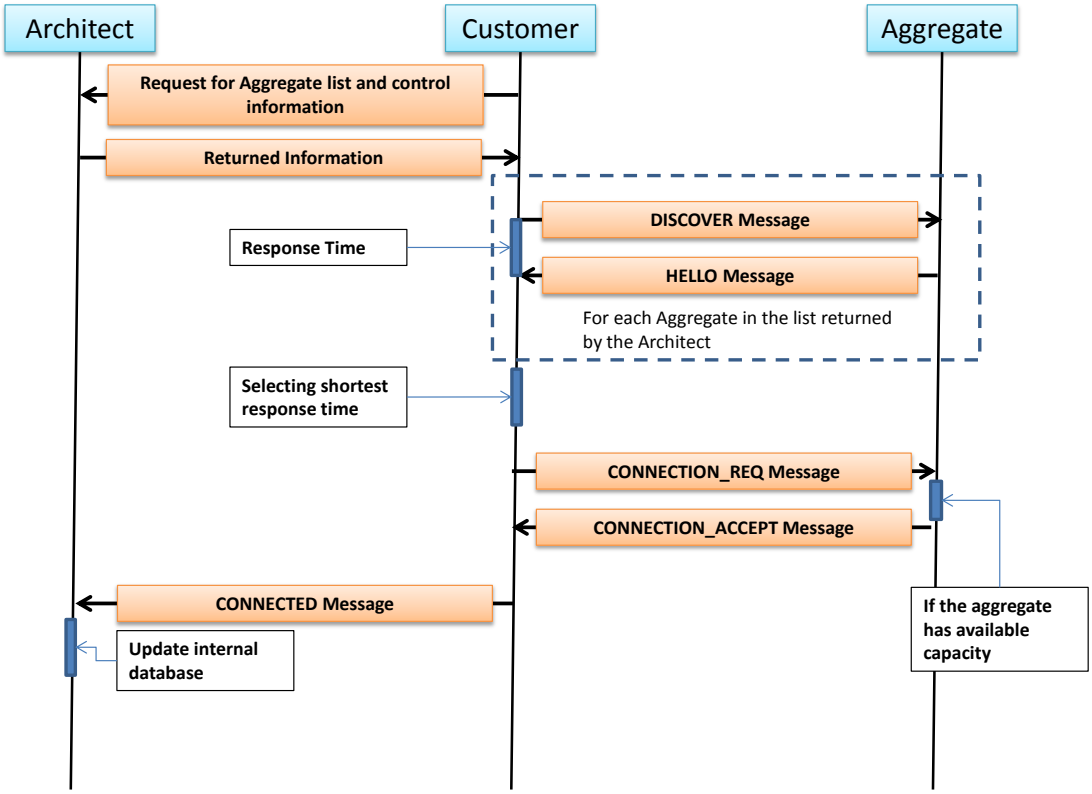


Fig. 5.1 – Initialisation Handshaking

If the first connection request was denied due to the aggregate having already accepted the maximum number of connections, the customer moved onto the aggregate with the second shortest response time until a connection is made. If no aggregates accepted the connection, the customer was then deemed to be isolated and communicates that to the Architect so that a place can be found, or additional aggregation resource is created. The same process was used to apply for a connection to a controller, if the aggregate layer was also performing the control processes, one connection request will apply to both data sharing and control. Furthermore if the level of control decentralisation is set to the customer layer, no control connection requests will be made as control will be handled locally.

5.2.1 Defining Substitute Agents

After both the data and control connections have been made – the final phase of the initialisation stage was the nomination of substitute agents. These agents are members of the customer/generation layer and were pre-selected to replace an aggregate in the event that the aggregate becomes unresponsive. Substitution was also one of the architecture transitions that could be selected by the Architect in the event of dealing with an error state which was having a large impact on a single aggregate within the network. An aggregate only selects a substitute once it had reached the maximum number of connections or, if no more connection requests were received indicating that agent population had been fully initialised.

To select a substitute the aggregate a data collection message was sent to each of the customers which had established communication connections, those customers then replied with a set of performance data including current input and output data flow rates, message congestion, and response times. From this information the aggregate could select the most appropriate agent to use at its substitute, in the event that the aggregate needs to be replaced for any reason the replacement had been preselected through the substitution process.

Once all customers and generation agents have secured their connections to the aggregate layer and a relevant controller and once substitution process had been completed for each of the aggregates the network can then be described as fully initialised.

5.3 PERFORMANCE MONITORING

The second stage of the process was the performance monitoring phase as the system needed to be monitored to ensure that it operated within a given set of parameters; the performance monitoring process was informed and adapted from work presented in [127] and [136]. In addition to recording data regarding differing communication and processing times, additional monitoring was performed through a range of key performance indicators (KPIs) documented in the following subsection.

In [127], the authors use the performance monitoring information to determine whether or not a fault has occurred, by applying threshold values to the performance monitoring metrics. If a metric exceeds a threshold a fault event is recorded, several fault events requires the intervention of a corrective mechanism. This approach was adopted into the

monitoring phase of the developed self-organising architecture, whereby a single metric exceeding a threshold is would not trigger an architecture transition.

A performance monitoring mechanism is considered to be one of the core functions of a self-organised system as presented by [37]. Furthermore the authors of [102] present a configuration whereby a specific agent or component is allocated for the purposes of collecting and processing of performance monitoring. Within the developed self-organising architecture individual agents perform local performance monitoring through comparing recorded data with a set of threshold values, however the overall processing of any performance data that did exceed one or more thresholds would be completed by a central agent. This was one of the responsibilities of the Architect agent which maintained a record of all performance exceptions and determined the overall system health.

Another contributor to the performance monitoring process was adapted from [118], where error reports have a limited lifespan – and once expire therefore the re-organisation mechanism would not be triggered by historic data. Furthermore the authors of [118] also presented a scenario whereby monitoring would be conducted locally and reported back to a central entity for processing. The use of the Architect has previously been stated but time limits on performance exceptions were also implemented such that the Architect did not attempt to respond to error conditions which had expired.

5.3.1 Threshold Decisions

One of the core methods behind the performance monitoring process was determining when each of the performance metrics was considered to outside of a recommended set of limits. Each agent was responsible for monitoring a series of performance metrics and reporting instances when those metrics exceeded performance limits. Therefore a set of thresholds needed to be instantiated to make the required comparisons; some thresholds were informed by practical limitations while others were informed through performing communication load tests to gauge effective difference between normal and abnormal data production and consumption. The following table presents the metrics examined and the thresholds applied

Table 5.1 – Threshold Settings Table

Metric	Threshold	Comment
Control Performance	400 Seconds	A 200 second waiting period was imposed before processing an excursion – if the event was not cleared

		with a 200 seconds of control being initiated a control error is triggered. The threshold of 200 seconds (plus the 200s standoff stage) is based on the voltage recovery times presented in [137] of 40-70s additional time is allowed for a non-optimal control approach.
Data Flow	Maximum Rate: 120kbps (15kB/s)	As per the Smart-meter Specification – smart-meters will use ZigBee standard transmitters and adhering to the ZigBee Smart Energy Profile (SEP) v1.2 [138]. The ZigBee devices have maximum potential transmission rate of 250kbps (31.25KB/s). Actual transmission speeds will be slower due overheads and delays. The selected threshold of 120kbps was in line with research conducted by the authors of [139] and [140]. The former identifying an effective maximum data rate of between 110kbps and 120kbps, while the latter observed performance loss after 118kbps. Therefore transmission would be possible after 118kbps, but with a performance loss. Additionally each ACL message within the MAS had its calculated size in bytes increased by 48 bytes to emulate the presence of a security certificate presented by the standard in [138].
Reactivity Response Time	500ms	The reactivity metric was set at 500ms as on the basis of testing the architecture without the presence of additional load to determine a baseline value. Lower thresholds were more likely to trigger false positives.
Congestion	50 messages	The congestion threshold was a measurement of the amount of messages that are waiting to be processed by the agent in question. Like the data flow threshold this primarily concerns the aggregation tier as these agents filter the bulk of the data and therefore form

		the effective data bottlenecks. Congestion was measured as a moving average over the course of a minute and therefore the threshold was taken from the average rather than congestion spikes. As with reactivity a set of base tests were performed to determine the nominal performance level given the agent population size, values that notably exceed that nominal level of congestion were the considered to be erroneous.
Unresponsiveness	No reply in 10 seconds. 10 missed messages	The unresponsiveness threshold was a limit by which transmitting agent assumes that the receiving agent is no longer responding to messages. If an agent did not receive a reply within 10 seconds sending the initial message, that message was declared missing. After 10 messages are declared missing the agent assumes that the desired recipient is offline or unresponsive
Under-usage	3kB/s	Research presented by the authors of [106] suggests that as part of Capacity and Coverage Optimisation in mobile communication networks, nodes which are under-utilised can be placed into a dormant state. This value is set at 20% of the maximum throughput rate – at 24kbps (3kB/s) This minimum utilisation of an agent, only applies to agents performing aggregation functions as they can be demoted or made dormant in response. Additionally customer or generation agents performing aggregation duties will monitor utilisation in the event that they need to relinquish aggregation responsibilities
Control Unresponsiveness	3 seconds	The control responsiveness metric was not used for performance monitoring purposes but the same techniques were applied. If a customer agent queried

		the controller to ask whether or not DSR restrictions are still required and receives no response within 3 seconds, the customer would terminate all control actions under the assumption that they are no longer needed.

Each of the performance metrics listed was stored as a performance monitor object, where samples were periodically taken so that a moving average could be computed. The performance monitor object also calculated the rate of change of a given metric so that if a threshold is exceeded, the object estimated the length of time the performance metric will remain outside the recommended limits and if it showed any sign of declining. Any metric that exceeds its threshold is considered to have entered an error state and thus the agent observing the error state would generate an error report and transmit that report to the Architect agent. The role of the Architect agent was to store the error with any other received reports and process the list with a view of determining the need for a transition event.

5.3.1 Error Collection and Processing

Processing and then acting upon an error state within the agent population was a three stage process. The first step involved handling the incoming error reports and storing them effectively – and thus required a suitable data structure to record all of the relevant error report fields. Step two required scanning through the error list and converting the data into a series of comparable metrics, error severities, totals, and scope. The final stage used the results of the error processing stage to determine if a self-organised action is warranted and if so which action to take.

The Architect agent would perform an analysis function whenever a new error was received from the agent population. At the point the message was received, the Architect determines whether or not an instance of the same error has been previously recorded. If it was the first instance of a given error then the agent will create a new error object and store the object in the list of active errors for that particular error type, however if the error has been received previously, the Architect would scan through the list of errors and update the relevant object with the new timestamp and magnitude of the most recent occurrence of the error. The list of active errors for each type is contained within a wrapper object; the wrapper object contains functions for processing the list of errors and returning information

necessary for performing decision making – I.E. prominent error locations or causes. For example if a customer observed a slow reaction time between itself its associated aggregate then it would send an error report. Each error report contains the following fields as illustrated in the following table in Table 5.2.

Table 5.2 – Composition of an example error alert message

Header	Error Type	Cause	Magnitude	Timestamp	Threshold
ERROR_ALERT	reactivity	AG1	750	1452790402508	500
Complete Message	ERROR_ALERT,reactivity,AG1,2000, 1452790402508,1500				

The Architect would receive this error and locate the wrapper object for the 'reactivity' error type, and extract the list of errors currently recorded for the error type. If it has been found that the same customer had previously reported a reactivity problem and cited agent "AG1" as the cause of the problem – the error report will be updated with the new magnitude and timestamp. If this was the first instance as previously indicated the Architect will create a new error report object. The following diagram outlines this process of handling incoming messages for any given error type.

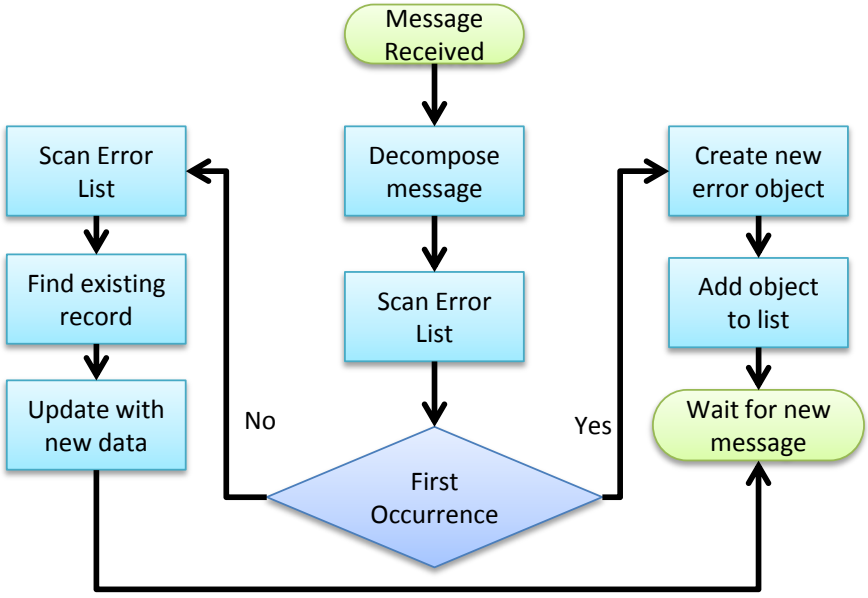


Fig. 5.2 – Receiving Error Messages

As part of the error management and performance monitoring stage it was also important for the list of errors retained by each of the wrapper objects to be processed – one of the most important parts of this process was to calculate error severities and rates of change. Error severity was determined as being the percentage by which the magnitude of each error exceeded the threshold value. The rate of change component would be extracted from error types with multiple error reports, as multiple data points were required to determine

if an error was decreasing or increasing in severity. Comparing the error magnitudes across several error reports were useful in determining how that error was developing – whether it was becoming more severe or whether it was declining. This information was then used as an input into the decision making process such that it could decide which errors were the most severe and which course of action was required.

Other factors are involved in the error processing stage, as it was important to filter out error reports which hadn't been updated in a significant period of time, if no further reports were made for an error condition it was then assumed that the error condition is no longer present. Therefore these errors were eliminated from the list of active errors as they were no longer active, errors are deemed to be inactive after 30 seconds of inactivity, and this prevented any expired errors from contributing to the average severity of the error type. This was also done to prevent error states carrying over from one transition event to another if they were solved in the first instance and avoid unnecessary transitions based around false positives.

At the end of the processing stage the Architect would then be aware if it needed to enact a transitional event through triggering the decision making element of the self-organising system. The elements of the decision making system will be documented in the following chapter, but the result of any decision was in the form of one of the following architectural transition options.

5.4 ARCHITECTURE TRANSISIONS

Four stages of architectural transition were available to the Architect agent in terms of adjusting the communication network in response to data retrieved from the performance monitoring stage. This data may indicate that the initialisation phase had created an uneven connection distribution across the aggregation tier, poor agent performance as a result of failure or the presence of an attack event within the architecture. Each stage represented an escalation in the severity of the response by the Architect agent and the severity of the performance issues across the agent population.

5.4.1 Stage 1 - System Rebalancing

A rebalancing operation was the smallest and least intrusive of the transition events and was based on systems used in mobile communication networks when redistributing connections between nodes. Rebalancing was most effective when redressing the actions

of the initialisation phase, where one aggregate may have been configured with a larger number of connected customer agents than any of other agents in the aggregation population. Therefore it was immediately at greater risk of congestion or reaction time errors, and as a result a rebalancing operation would transfer some of the connections from the heavily loaded aggregate to a less heavily loaded equivalent. A second format of the rebalancing stage took place when a small number of agents are affected by connectivity issues between themselves and the aggregation layer in this case those agents were transferred to a new location.

Process

If the Architect agent determined that a rebalancing process was required it would trigger the *transferConnections* function – the pseudocode to which is presented in Fig. 5.3. The function was supplied with an error type this error type represented the dominant issue detected during performance monitoring, if the dominant error referred to connectivity problem, either in the form of slow response times or unresponsiveness – a targeted rebalancing is triggered. Other error formats such as congestion or execution timing errors are responded to with a general rebalancing.

```
FUNCTION CALL with dominant error type
INITIALIASE list of connections to transfer
IF error type = “reactivity” or error type = “unresponsive”
        RETRIEVE list of agents reporting the error
        FOR agent list
                RETREIVE the aggregate the agent currently reports to
                FOR each of the potential aggregates
                        IF aggregate has a remaining connection space
                                BUILD transfer message
                                SET content = “NEW_TARGET”
                                SET recipient as agent to be relocated
                                SEND transfer message

                                BUILD transfer out message
                                SET content = “TRANSFER_OUT”
                                SET recipient as former aggregate
                                SEND transfer out message
                                RECORD all message sizes in bytes
                                END FOR
                        ELSE
                                MOVE to next aggregate
                        END IF
                END FOR
        END FOR
ELSE
        IF aggregate reporting the error has the fewest connections
                PRINT “Rebalancing would be ineffective, replacing agent
                FUNCTION CALL to substitution function
```

```
        ELSE IF multiple aggregates reporting errors
                FUNCTION CALL activate a single dormant agent
        ELSE
                CALCUATE difference in connections between error reporting aggregate and least
                heavily loaded aggregate
                TRANSFER half the difference in connections to the least heavily loaded
                aggregate
        END IF
END IF
```

Fig. 5.3 – Pseudocode Regarding Connection Rebalancing

The pseudocode also demonstrates that the Architect has the power to over-ride the suggestion of a rebalancing action if the location of the problem determined that such an action would prove to be ineffective. If the error was located at an agent which had the

fewest connections, it indicated another issue with the agent itself rather than one derived from communication volume and therefore would trigger a stage 2 substitution action instead. Alternatively the Architect may discover that the issue was widespread across a number of locations and therefore rather than redistributing the connections via a rebalancing action – it would be more advantageous to increase the aggregation capacity by activating a dormant agent.

Communication Summary

During a rebalancing operation the following messages would be transmitted between the Architect, aggregation and customer layers of the architecture. Each of the rebalancing formats utilised the following three messages presented in Table 5.3 to orchestrate transferring a connection from one aggregate to another.

Table 5.3 – Communication Summary during Rebalancing.

Header	Sender	Receiver	Payload	Comments
NEW_TARGET	Architect Agent	Customer or Generator Agent	Agent name of the new aggregate to connect to	Message informing the customer or generator it is being transferred and to modify its communication targets.
TRANSFER_IN	Customer or Generator Agent	New destination aggregate	Header only	Message to introduce the customer/generator to the new aggregate agent
TRANSFER_OUT	Architect Agent	Former data aggregate	Header only	Message to the aggregate formerly connected to the customer agent to removed it from its connection list

5.4.1 Stage 2 - Agent Substitution

Aggregation level substitutes were pre-selected during the initialisation stage, as adapted from the fault tolerant aspects of the EDETA [114] method. The role of a substitution event was to replace an existing aggregate with an immediate substitute agent. Substitution was also presented in [132] whereby local controllers which fail are replaced by an agent from a lower tier in the hierarchy. This process would not add to the total aggregation capacity available in the agent population, it replaces a poorly performing or failed agent with another functioning one. Because the replacement agent was from a lower tier, it maintained its existing responsibilities (monitoring local voltage, publishing demand profiles, enacting any control commands etc.) while also adopting those of the failed agent.

The substitution process used some of the same functions as used by the promotion process, but without adding aggregation capacity or modifying network tiers

Process

When a substitution was called for as part of an agent failure the process will remain the same, only the trigger condition changes. So the first stage was to identify which of the connected customer/generator agents was the designated substitute for the aggregate in question. This designated substitute was then sent an "ACTIVATE_SUB" command, and supplied with a list of agent names which the substitute would assume the responsibility of interacting with. As the substitute agent received the message, it began informing the agents who were formerly connected to the failed aggregate who to redirect their messages to. As outlined in the pseudocode listed below in Fig. 5.4 :

```
RECEIVE "ACTIVATE_SUB" Message and customer list
SET aggregate status = true;
BUILD agent message declaring aggregate status
FOR each customer in customer list
        ADD customer id to list of message recipients
END FOR
SEND message
RECORD size of outgoing message in bytes
```

Fig. 5.4 – Pseudocode for Activating a Substitute

The substitute would then break down the list of affected customers from the message into an array structure so that it can iterate through the list and transmit the command to each agent to inform it of the change in source of aggregation services. The substitute performed this through transmitting a "SUB_ACTIVE" message to each member of the agent list. On reception of this message, other agents previously associated with the failed aggregate reset their *data_target* which defined the outgoing location of agent updates. The substitute itself would modify its own *data_target* to point to the Observer agent if anything other than the tiered architecture is in use. If a tiered architecture was in place, the substitute would connect with an upper tier aggregate instead. Once all the agents involved in the process have reset their communication targets the broken link will have been repaired as the failed or inaccessible aggregate is phased out of the communication loop as the substitute assumed responsibility.

Because the substitute did not participate in the original initialisation phase as a substitute agent it will not have built internal knowledge base containing information about those connections. Therefore to accumulate that information, the agent identifiers extracted from incoming demand and generation updates were used to build the relevant internal knowledge of connected agents. When a new the first update from a connected customer is received, sender information was retrieved from the message, such that it could be added to the list of customers which relied on the aggregation services provided by the replacement aggregate. The following pseudocode in Fig. 5.5 outlines the function which handled adding connection objects and determining the object type based on the first letter of the agent name – where "C" refers to a customer agent and "G" refers to a generator agent.

```
FUNCTION CALLED with new connection name
FOR each customer in connection list
        RETREIVE customer name
        IF new customer name matches current customer name
                RETRIEVE customer object
        END IF
END FOR
IF no matches
EXTRACT name prefix ("C" or "G") from new connection name
BUILD new customer object
IF prefix = "C"
        SET object type to "CUSTOMER"
ELSE IF
        SET object type to "GENERATOR"
END IF
ADD customer object to connection list
RETURN customer object
```

Fig. 5.5 – Pseudocode for Checking and Adding Connection Objects

Both agent types were represented by the same java object as they both contain the following variables: agent name, type, real power (load or output) and reactive power (load or output). The role of the agent type distinction was to determine how the information was aggregated – to separate P and Q for load and generation entities.

Communication Summary

The following table presented in Table 5.4 documents the series of messages involved in delivering a substitution process as the architect activated the substitute and ensured that

data from the outgoing aggregates isn't used in overall calculations to avoid data duplication.

Table 5.4 – Communication summary for Agent Substitution

Header	Sender	Receiver	Payload	Comments
ACTIATE_ SUB	Architect Agent	Substitute Agent	Substitution type, Communication architecture type, List of agents to transfer	Message from the architect to a substitute agent instructing it to activate and which customers to connect to
DROP_ AGGREGATE	Architect Agent	Observer Agent	Name of aggregate being replaced	Message to the observers agent to inform it that one of the aggregates previously connected was being replaced
SLEEP	Architect Agent	Aggregate being replaced	Reason for being made dormant "SUBBED"	Message to the aggregate being replaced by the substitution instructing it to enter a dormant state
SUB_ACTIVE	Substitute Agent	Customers being reconnected	Reason for transfer "SUBBED CUSTOMER"	Message to each customer formerly connected to the replaced aggregate informing them that the substitute is the new aggregate
SUB_ COMPLETE	Customers being reconnected	Substitute Agent	Header only	Response from the customers being transferred that they

				have accepted the substitution and changed internal parameters accordingly.
IS_ AGGREGATE	Substitute Agent	Observer Agent	Header only	Message to inform the observer that the substitute agent will now be acting as an aggregate

Results

To illustrate the system in effect Fig. 5.6 presents the change in incoming and outgoing data before and after a substitution event. The solid lines represent the data flow of the original aggregate before it is taken out of the loop, while the dashed lines represent the data flow of the substitute agent. In the pictured example the substitution event was hard coded into the architect agent and triggered using a timer rather than a response to network conditions. This process was applied to examine the effectiveness of the transition event and demonstrate that the processes and communication structure previously presented would have the desired impact. These hard coded triggers were then replaced when installing the decision making engine into the architect agent, once the individual transitions were validated in isolation of the error recording processes.

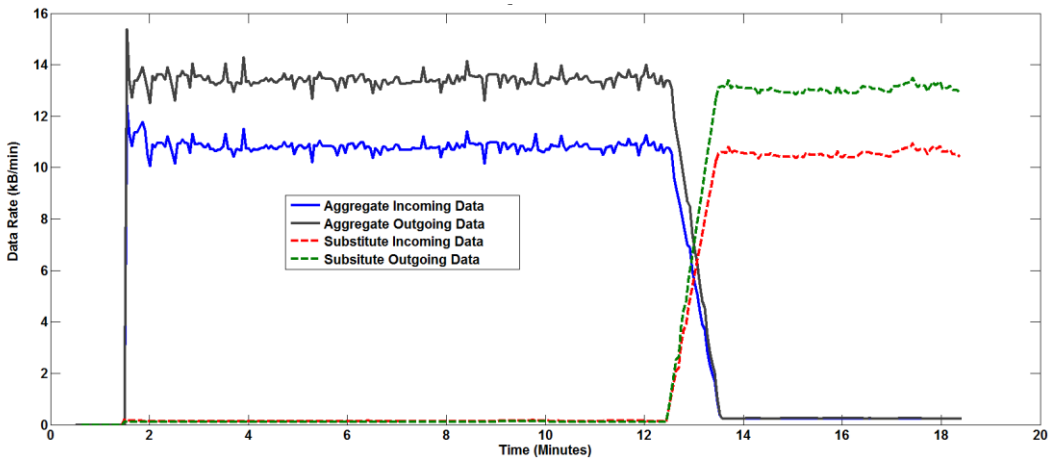


Fig. 5.6 – Data Transfer during Substitution

Twelve minutes into the simulation the changeover procedure was triggered and at 13 minutes the dataflow recorded by the substitute matches that of the former aggregate. The

minute delay between the two samples was due to the performance monitoring process collecting data flow information which is sampled over a minute period. So a full minute after the transfer the array containing performance monitoring data was updated with the results of taking over the aggregation role.

5.4.2 Stage 3 - Activating Dormant Agents

The second mechanism involved the activation of dormant agents, a set dormant aggregation layer agents existed to provide additional aggregation capacity in the event that performance monitoring detected a wider issue. This would indicate that the aggregation layer had become over-loaded and further aggregation resources are needed to be dispatched to distribute the connections. In contrast to the substitution process, all existing agents retain their original responsibilities and the architecture remains largely consistent, no further tiers would be added and no roles modified. The dormant agents are activated and assigned to the lower or only aggregation tier and communicated with the existing aggregates. Likewise if the congestion/data flow information indicates that the agents are being under-utilised then they can be placed into a dormant state and the connections re-allocated to the remainder of the aggregation population.

The use of dormant agents in self-organisation was documented by the authors [106] – whereby nodes with little or no connection usage are placed into a dormant state and are called upon when needed. Additionally several sensor network solutions place agents into a dormant state through for the purposes of conserving energy, however as it is not anticipated that the agents will have a limited power supply – a dormant agent remains partially active in the sense that it can listen and respond to messages. Such that if it received errant communications while being registered as dormant, an appropriate message can be send in response, also to allow the agent to listen for signals to awaken. Nodes which were not used as data collection points were also made dormant in the case of the Tic-Tac-Toe-Arch [111], as part of the initialisation phase. In addition to a series of aggregation agents which were instantiated as dormant agents, any active aggregates which did not receive any connections during initialisation would themselves enter a dormant state.

Process

If the Architect agent determined that the appropriate course of action was to activate members of the dormant agent population the first step was to trigger the *activateDormants()* function within the Architect agent. This function checks the number of dormant agents,

and then checks the number of active aggregates. Fig. 5.7 presents the pseudocode for the function responsible for activating the dormant agents an integrating them with the rest of the network.

The first act of the function is to determine how many connections will the newly awakened aggregate be responsible for, and which member of the active aggregate population those connections would be transferred from. To calculate the number of customer connections to relocate, the architect needs to be aware of the ratio between active aggregates and dormant agents scheduled to be awakened. For example if a structure with 4 active aggregates is to be joined by 2 dormant agents – the first dormant agent would receive 33% of the connections from both the first two active aggregate. The second dormant agent would also receive the same proportion of customers but from the second pair of aggregates – therefore all of the active dormant agents would adopt a similar number of connections and communication load at all aggregates was reduced.

```
FUNCTION CALL
RETRIEVE list of active aggregates
RETRIEVE list of dormant aggregates
SET limit = ratio of active to dormant aggregates

SET counter = 0
IF dormant aggregates are available
        FOR each dormant aggregate
                IF (number of active aggregates – counter) < limit
                        SET remaining active aggregates = number of active aggregates – counter
                END IF
                FOR each of the remaining set of active aggregates
                        ADD aggregate name to transfer list
                        Increment counter
                END FOR
                BUILD activation message
                SET recipient as current dormant aggregate
                SET content as: List of agents to take connections from, number of connections
                to take
        END FOR
END IF
```

Fig. 5.7 – Pseudocode for Activating All Dormant Agents

Once the new distribution of customer connections was calculated the "ACTIVATE_DORMANTS_ALL" message would be transmitted to each of the dormant

agents. This message included a list of original aggregates to request connection transfers from, and the proportion of agents to request. The pseudocode presented in Fig. 5.8 presents the reaction by a dormant aggregate to the command to activate and participate in the network. The receiving agent would initially check that is it in a dormant state, to ensure that the selected dormant agent had not previously been activated by a different transition event. Once passing this check the dormant agent compiled two messages – the first of which was sent one back to the Architect as a confirmation that the agent is now active, this allowed the Architect to amend the number of dormant and active agents it is aware of for future reference. The second message was sent to the set of aggregates whose names were supplied as part of the activation call, this message informed the selected aggregates that the dormant agent is now active and will accept connection transfers.

```
RECEIVE activation message, and list of targets to transfer customer connections from
BUILD reply message
IF aggregate is dormant
        SET dormant = false
        SET reply content = "NOW_ACTIVE"

        BUILD transfer message
        SET transfer message content = "DORMANT_ACTIVE"
        FOR each transfer target
                ADD transfer target name to transfer message
        END FOR
        SEND transfer message
        RECORD message size in bytes
ELSE IF aggregate is not dormant
        SET reply content = "ALREADY_ACTIVE"
END IF
SEND reply message
RECORD message size in bytes
```

Fig. 5.8 – Dormant Agent Receiving the Call to Awaken

The message also indicated what proportion of current connections should be transferred to the new aggregate. Fig. 5.9 describes how an active aggregate agent responds to the announcement of an awoken dormant agent.

```
RECEIVE dormant activation message, number of connections to transfer
BUILD transfer message
SORT connection list
FOR number of connections to transfer
        RETRIEVE connection from list
        SET connection name to transfer message recipients
        REMOVE connection from list
        REDUCE connection total by one
END FOR
SET transfer message content to "NEW_TARGET"
SEND transfer message
RECORD message size in bytes
```

Fig. 5.9 – Active Aggregate Becoming Aware of an Awoken Dormant

Upon receiving a "DORMANT_ACTIVE" message the aggregate agent the extracts the parameter containing the proportion of agents to transfer. It would iterate through the connection list and send a "NEW_TARGET" message to a correct proportion of the connected customers/generators. This "NEW_TARGET" message contained an identifier for the newly activated dormant agent which was accepting transferred connections. Fig. 5.10 outlines the process that a customer or generator agent goes through when in reception of a transfer notification.

```
RECEIVE new target message and id of new aggregate agent
IF aggregation control in place
        SET controller id to data target id
END IF
IF is designated substitute agent
        BUILD replace substitute message as reply
        SET content "NEW_SUB"
        SEND reply message
        RECORD message size in bytes
END IF
SET data target = new aggregate id
BUILD transfer complete message
SET receiver as new aggregate id
SET content as "TRANSFER_IN"
SEND transfer complete message
RECORD message size in bytes
```

Fig. 5.10 – Receiving a Transfer Notification

Before a customer/generator can transfer to another aggregate agent would have to conduct two checks which determine how it relates to the aggregate agent it was being transferred

from. The first check was to determine which control format the architecture was operating under – if the aggregation tier was performing control then the data target and the control target point to the same agent. If a customer was being relocated to another aggregate, it also needed modify its control target such that control alerts are also redirected to the former dormant agent which assumed control responsibilities.

The second check was to determine if the customer agent was selected to be a substitute for the aggregate it is being disconnected from, if that is the case the aggregate needed to be informed such that it can select a new substitute to replace it. This is done through sending a "NEW_SUB" message back to the original aggregate which triggers a new substitute selection process. Finally the customer/generator sends a "TRANSFER_IN" message to the new aggregate so that it can be added to its connection list.

Communication Summary

The following is a list of the messages involved in the process of activating a dormant agent and transferring a set of connections from existing aggregates to the newly awakened dormant aggregate. Each message used the ACL message protocol – and was composed of two sections. The first section was the header which was used by the listening behaviour, which acted as a function switchboard for the agent. The second was the payload and contained any variables and information that the receiving function needed to perform the required action.

Table 5.5 – Communication Summary

Header	Sender	Receiver	Payload	Comments
NET_INFO	Gateway agent (DMA)	Architect	Number of customers Number of Generators List of Generator names Number of Dormant aggregates List of Dormant Aggregates	Informs Architect about the contents of the components CSV file including aggregate agents

			Number of Active Aggregates List of Active Aggregates	
ACTIVATE_ DORMANTS_ ALL	Architect	Dormant Aggregate	List of active aggregates to contact Proportion of connections to transfer	The command is sent to all dormant agents in this case rather than activating a single agent in one part of the network
NOW_ACTIVE	Dormant Aggregate	Architect	Header only	Confirmation message that the activation has taken place
ALREADY_ ACTIVE	Dormant Aggregate	Architect	Header only	Message to inform the Architect that this agent has already been activated
DORMANT_ ACTIVE	Dormant Aggregate	Active Aggregate	Proportion of connections to transfer	This initiates the connection transfer process
NEW_TARGET	Active Aggregate	Customer or Generator Agent	Agent name of the new aggregate to connect to	Message informing the customer or generator it is being transferred and to modify its communication targets.

NEW_SUB	Customer or Generator Agent	Active Aggregate	Header only	Message to inform the aggregate that its substitute is being transferred and to search for a replacement
TRANSFER_IN	Customer or Generator Agent	Former Dormant Aggregate	Header only	Message to introduce the customer/generator to the new aggregate agent

Results

The dormant activation process was tested with 340 customers 4 generators initially connecting to four aggregation agents these agents did not make symmetrical connections in the sense that each aggregate did not receive an equal number of connections. During a test simulation two additional aggregates which previously existed in a dormant state only, were activated and instructed to initiate the transfer of customer data connections from the original set of aggregation agents. The following graph in Fig. 5.11 traces the number of connections per aggregate throughout the simulation runtime.

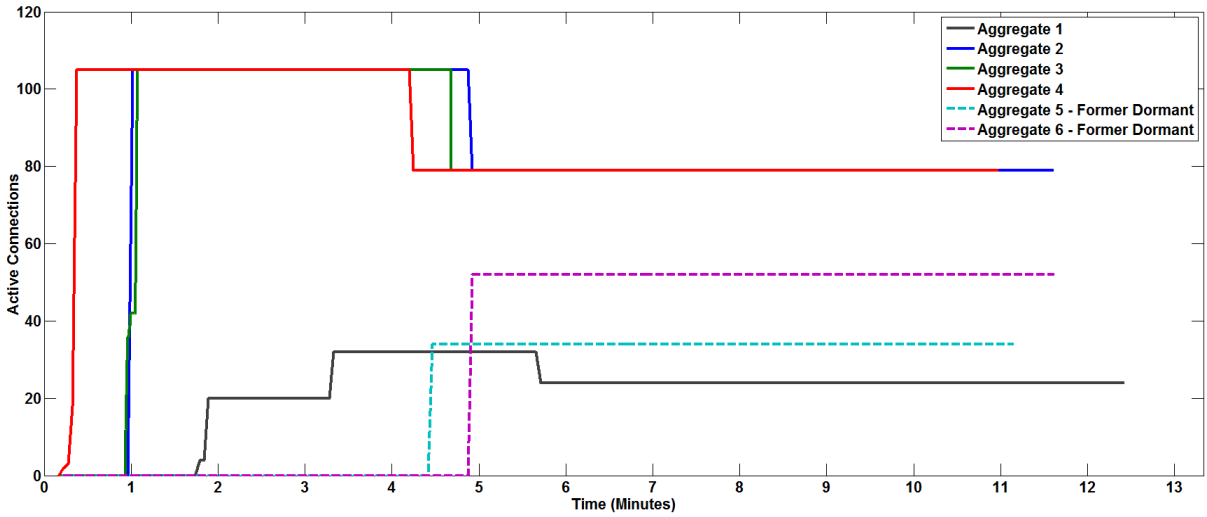


Fig. 5.11 – Changes in Number of Connections per Aggregate

The graph shows that aggregates 2-4 reached their soft connection limit reasonably quickly and therefore the remaining customers discovered and connected to AG1 explains the delay

in accumulating connections. When the dormant agents (AG5 and AG6) were activated, they took connections away from the original set of aggregates. Those aggregates which were up against the soft limit shed more connections than AG1 which was initially configured with fewer connections. It can also be seen that the two newer aggregates did not accumulate the same number of transferred connections because of the respective population sizes of the aggregates they are transferring connections from. For example AG5 received 1/3 of the connections of AG1 and 1/3 of the connections from AG2 – as AG1 had fewer connections AG5 gains fewer connections. Whereas AG6 retrieved connections from two of the highly populated aggregates and therefore is handed a larger population itself.

5.4.3 Stage 4 - Promoting and Demoting Agents

The promotion mechanism operated in a similar manner to the substitution process, with the exception that customer/generation agents were promoted for the purposes of increasing the aggregation capacity rather than to replace an existing agent. Also aggregates could be promoted to create an additional aggregation tier to reduce the load on the observer agent. When creating an additional tier, each of the promoted aggregates is replaced by a promoted customer or generator agent. Selecting agents to replace the promoted aggregates take advantage of the substitution process.

The concept of promoting agents was connected to the EDETA approach in the sense that cluster-head agents are promoted from the overall agent population, as all sensors are considered to be homogenous and it was their physical location which forms the network. A select number of standard sensor nodes are elected to take up the role of cluster-heads, and in the event that the population of cluster head nodes doesn't provided sufficient coverage, additional sensors are promoted to assume the role. Another example of promotion and demotion of agents from a lower tier into a higher one is presented in [141]. In this example matchmaking nodes pair producer and consumer nodes together, if the matchmaking process becomes too heavily loaded additional resource is created through promoting producers or consumers to become matchmaking nodes. This process was similar to the current promotion process adopted by the self-organising architecture – customers/generators are promoted to assist in the aggregation layer if required.

Two forms of agent promotion have been developed as latter stage architecture transition option, the first of which was to promote an aggregate agent up to a higher tier to move from a base or clustered architecture to a tiered alternative. The second was to create

additional aggregation resource in the lower aggregation tier – in the event that no dormant agents are available or if previous activation of dormant agents has not been enough. The second approach was not necessarily considered to be a stage four transition, but one that was introduced as an alternative mechanism when the selected transition stage could not be executed.

Promotion for a tiered architecture

To trigger a promotion event for the purposes of creating an upper tier, a function within the Architect agent would be called – as in the same manner which other transitional procedures will be triggered. When applying the tiered architecture across the network/zone under control the Architect had to initially select which of the current aggregate population was going to be promoted into an upper tier. This decision was based on the data collection process that the Architect agent periodically carries out of the aggregate population – therefore recent information on congestion was collected, data flow and reactivity to the observer. Such a transition would be triggered in the event that the observer noted an increased quantity of information being passed to it by the aggregation layer or that reaction times between the top two tiers of the architecture are starting to rise.

The first stage was for the Architect to query the aggregate population and retrieve a list of those aggregates identified as being in the lower aggregation tier as per the extended *getList()* function. The Architect agent would choose one or more aggregates to promote into the upper tier. From the total aggregation population, those agents with the highest congestion figure were selected, a higher congestion value were used because this reflects the aggregates which are experiencing the highest load in their current role and therefore would benefit from promotion into a less highly loaded tier. The selected aggregates are then contacted with a "PROMOTE_SELF" message. This function is presented in the following pseudocode in Fig. 5.12, which also includes the Architect informing the substitutes for the promoted aggregates to activate and replace the aggregate within the lower aggregation tier.

```
FUNCTION call with number of agents to promote (n)
RETREIVE list of lower tier aggregates
FOR each lower tier aggregate
        IF promotion list size < number of agents to promote
                RETREIVE current aggregate queue size
                COMPARE value to maximum queue size
                IF found largest queue size
                        ADD aggregate to promotion list
                END IF
        END IF
END FOR
IF promotion list not empty
        BUILD promotion message
        SET content = "PROMOTE_SELF"
        FOR each promotion candidate in list
                ADD candidate to recipient list
        END FOR
        SEND promotion message
        RECORD message size in bytes
ELSE IF
        PRINT no promotion candidates found
        RETURN transition incomplete
END IF
```

Fig. 5.12 – Triggering Aggregate Promotion

Upon receiving a "PROMOTE_SELF" message from the Architect, each of the promoted aggregates sets their data target to communicate with the observer, in the event that a previous promotion phase had changed the data target variable. Additionally it sets the current communication architecture to "tiered" and responds with a "PROMOTE_CONFIRM" message to inform the Architect that it received and acted upon the command. Once the Architect had received confirmation messages from each of the promoted aggregates, it then informs the lower level aggregates that a tiered architecture has been created and that they are now members of a lower tier. As illustrated in Fig. 5.13

```
RECEIVE Promotion confirmation message
INCREMENT promotion count tally
SET sender's aggregation tier status
IF promotion tally = number of intended promotions
        SET communication Architecture = tiered
        RETRIEVE list of lower tier aggregates
        BUILD lower tier message
        SET message content = "LOWER_TIER"
        FOR each lower tier aggregate
                ADD aggregate id to recipient list
        END FOR
        SEND lower tier message
        RECORD message size in bytes
END IF
```

Fig. 5.13 – Informing the Lower Tier

Because the lower tier aggregates do not have direct access to the observer as they previously did, these aggregates then need to select a connection from the upper tier. This process mirrored the initialisation phase for customer/generation agents – whereby the aggregate is given the names of all upper tier connections and contacts them all to retrieve round trip time information. The lower level aggregate is also required to contact the observer to inform it that it will no longer be directly publishing updates, so that it can be removed from the observer's data collection structure to prevent information duplication. Once the lower level aggregates were connected to the upper tier – all agents will once more be connected and the process is complete. The promoted aggregate was then replaced by its own substitute and, the remainder of the set of aggregates seek out upper tier connections.

Promotion for additional aggregation resource

The second avenue of agent promotion related to increasing the amount of aggregates within the single lower aggregation tier. This was similar to the process involved with activating dormant agents, to back-up the current set of aggregates. However if no dormant agent population was present or those dormant agents had already previously been contacted and activated, an alternative solution is to use members of the customer/generation layer as aggregation agents.

Initially this process was triggered by the Architect agent, whereby "PROMOTE_CUSTOMER" message was transmitted to each of the currently active aggregation agents. This effectively doubled the aggregation resource by assigning each of

the current aggregates a promoted back-up to share the connection load with. This function is presented in Fig. 5.14 – and was received within the listening behaviour of the target aggregate, as presented in Fig. 5.15. Once the aggregate agent received the command to promote one of its customer agents up to aggregate status – it uses the same process that the Architect uses to select aggregates for promotion up to an upper tier. It uses the data collected for each of the agents and selects the agent with the longest round-trip time, as this agent could be seen as the least well connected and would be reasonable to be selected as a new aggregate.

```
FUNCTION CALL
RETRIVE list of active aggregates
BUILD promotion message
SET content = “PROMOTE_CUSTOMER”
FOR each active aggregate
        ADD aggregate id to recipient list
END FOR
SEND promotion message
RECORD message size in bytes
```

Fig. 5.14 – Promotion into the Lower Tier

```
RECEIVE promotion message
RETRIEVE connection information for customer with longest communication round trip time
BUILD promotion message
SET content = “PROMOTE_SELF”
ADD customer id to recipient field
SEND promotion message
RECORD message size in bytes
```

Fig. 5.15 – Receiving a "PROMOTE_CUSTOMER" Message

The least well connected aggregate was selected as it represented an agent which may be more geographically distant from the current aggregate. Therefore the process involved promoting a customer agent into an aggregation potion where little aggregation was present.

The selected customer receives the “PROMTOTE_SELF” message from the Architect, and checks whether or not it is a substitute agent – if that is the case it won’t accept the promotion request. Otherwise the agent will transmit the promotion confirm message to the aggregate, as well as contacting the Architect and observer agent to inform them of the role change so that their data structures can be modified to recognise the change. The response from the perspective of the promoted customer is presented in the pseudocode extract in Fig. 5.16

```
RECEIVE promotion message
IF is not a designated substitute
        SET aggregate status = true
        BULD reply message
        SET content = "PROMOTE_CONFIRM"
        SEND message
        RECORD message size in bytes
        BUILD aggregate status message
        SET content = "IS_AGGREGATE"
        ADD Architect and observer agents to recipient list
        SEND message
        RECORD message size in bytes
        IF communication architecture is not tiered
                SET data target to observer agent
        END IF
ELSE IF designated substitute agent
        BUILD reply message
        SET content = "PROMOTE_DENY"
        SEND message
        RECORD message size in bytes
END IF
```

Fig. 5.16 – Response to Receiving a "PROMOTE_SELF" Message

Once the promotion was complete and each aggregate has been informed of the successful promotion event, it was then responsible for transferring a series of connections from its own connection list to the promoted customer. Each aggregate agent transfers 50% of its connections to the promoted customer. This is illustrated in Fig. 5.17, whereby a series of "NEW_TARGET" messages to agents which were to be transferred.

```
RECEIVE promotion confirm message
SORT connection list
BUILD connection transfer message
SET content = "NEW_TARGET"
FOR half of the connection list
        ADD connection id to recipient list
        REMOVE connection from connections list
        DECREMENT connection total
END FOR
SEND transfer message
RECORD message size in bytes
```

Fig. 5.17 – Transferring Connections

Once all the agents have been transferred, the process of adding a customer in the place of an additional aggregate is complete. The promoted customer will then aggregate incoming updates and pass the totals onto the observer agent in the presence of a single tier of aggregation agents. In a tiered architecture those updates are transmitted to the upper aggregation layer.

Communication Summary

The following series of messages presented in Table 5.6 were used in the process of promoting agents, both in terms of increasing the number of aggregates in the lower tier, but also in terms of adding a second aggregation tier to the network.

Table 5.6 – Promoting Aggregates.

Header	Sender	Receiver	Payload	Comments
PROMOTE_ SELF	Architect	Selected Aggregates	Header only	Command for an aggregate agent to move to an upper tier
PROMOTE_ CONFIRM	Selected Aggregate	Architect	Header only	Confirmation that the aggregate has been promoted
ACTIVATE_ SUB	Architect	Designated Substitute	Current time Substitution type (PROMOTE) Communication architecture Set of agents to transfer	Instruction to the substitute of the promoted aggregate to take over the role
SUB_ACTIVE	Designated Substitute	Architect	Header only	Confirmation that the substitution occurred
SUB_ACTIVE	Designated Substitute	Customer Agents	Header only	Instructing the other customers

				that the substitute is the new aggregate
SUB_COMPLETE	Customer Agents	Designated Substitute	Header only	Confirmation that the customer has changed its data target
IS_AGGREGATE	Designated Substitute	Promoted Aggregate	Header only	Informing the upper tier that the agent is performing as an aggregate
LOWER_TIER	Architect	Lower tier Aggregates	List of upper tier aggregates	Information message to the lower aggregates informing them of a tiered structure
LOWER_TIER	Lower tier Aggregates	Observer	Header only	Informing the observer that the lower aggregates will not be contacting it
DISCOVER	Lower Tier Aggregates	Upper Tier Aggregates	Time message sent	Initial discovery message sent to upper tier aggregates
HELLO	Upper Tier Aggregates	Lower Tier Aggregates	Time original message sent Time message received	Response to discovery messages

			Time reply sent	
JOIN_REQUEST	Lower Tier Aggregates	Upper Tier Aggregates	Time message sent "AGGREGATE" flag	Request for connection
JOIN_CONFIRM	Upper Tier Aggregates	Lower Tier Aggregates	Time original message sent Time message received Time reply sent	Connection request approved

Table 5.7 – Promoting Customers

Header	Sender	Receiver	Payload	Comments
PROMOTE_ CUSTOMER	Architect	Lower Tier Aggregates	Header only	Instruction to start a promotion process
PROMOTE_SELF	Lower Tier Aggregates	Selected Customers	Header only	Instruction from the aggregate to promote one of its connected customers
IS_AGGREGATE	Selected Customers	Observer, Architect	Header only	Inform both the Architect and the observer that the customer is behaving as an aggregate
PROMOTE_ CONFIRM	Selected Customers	Lower Tier Aggregate	Header only	Confirms to the aggregate that promotion is complete

NEW_TARGET	Lower Tier Aggregates	50% of connections	Id of the promoted customer agent	Instructs some of the connected customers to connect to the promoted agent
NEW_SUB	Transferred customers	Lower Tier Aggregates	Header only	Informs the aggregate that it's substitute has been transferred and to select a replacement
TRANSFER_IN	Transferred customers	Promoted customer	Agent type flag	Transfers connection from the old aggregate to the promoted customer

Results

To demonstrate the impact of a promotion event the following figure presented in Fig. **5.18**, where two agents are promoted in support of a pair of aggregates. The two promotion events were aimed at agents with differing communicative loads to illustrate the impact on both a high and low functioning aggregate.

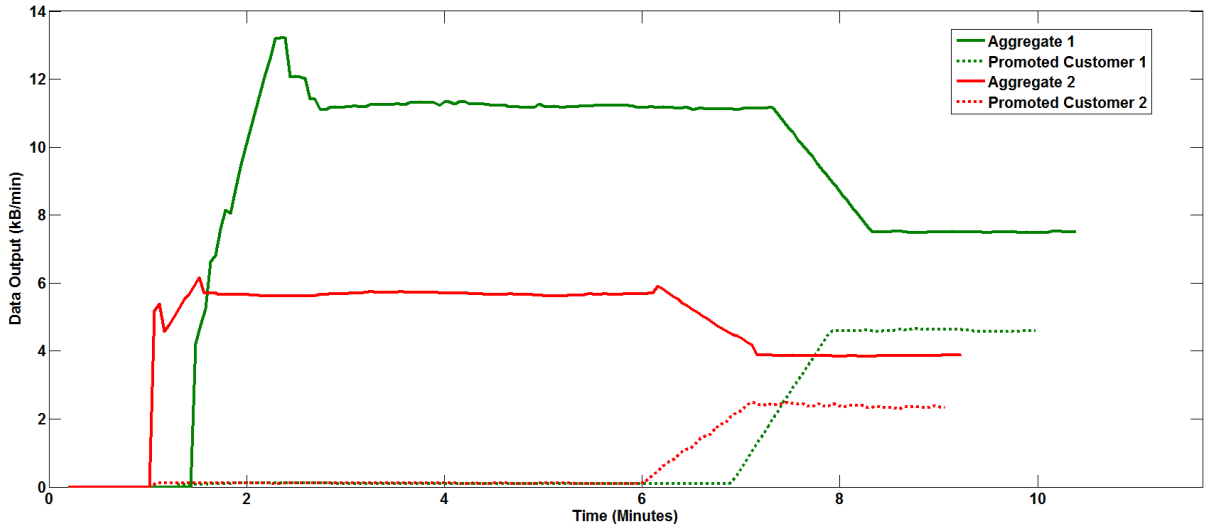


Fig. 5.18 – Data flow during a Promotion Event

At the point where the first agent was promoted, the original aggregate observed a reduction in data flow; this was picked up by the promoted customer agent as it assumed a proportion

of the connections previously held by the aggregate. The same pattern was present with respect to the second of the promotion events as the losses experienced by the Architect agent were collected by the promoted customer, such that no connections are dropped during the process. This was further documented in the following figure, illustrating the number of customer connections managed by the agents involved in the process

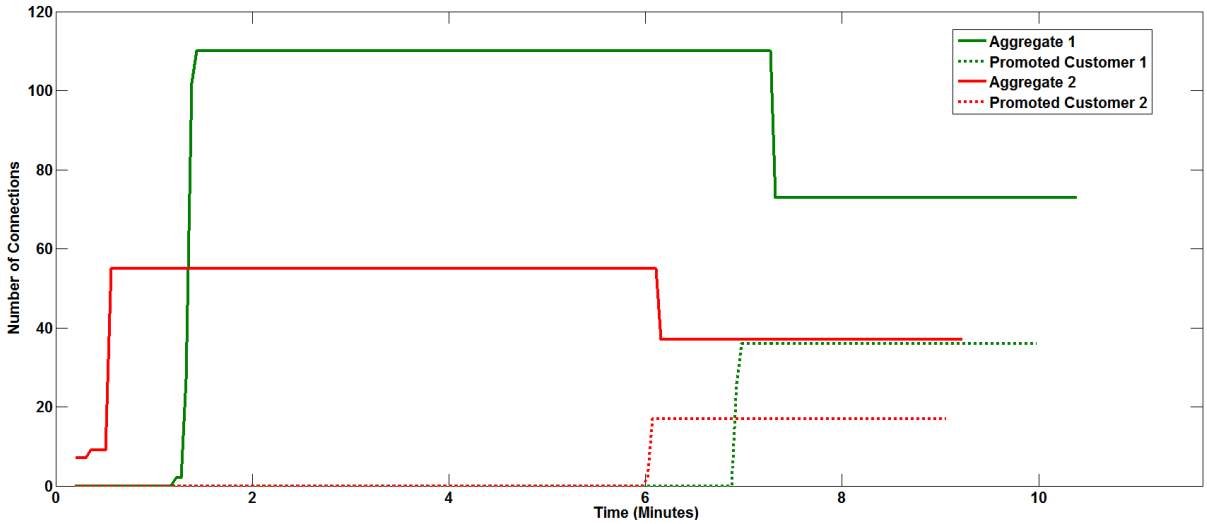


Fig. 5.19 - Number of Connections per Agent during a Promotion

Both figures represented a promotion scenario where the additional aggregation resource in the form of customer agents was added to a single aggregation tier focussing on the process of converting conventional customer agents into aggregation roles.

5.5 SIMULATING ATTACK EVENTS

The process of implementing the attacks was embedded in the development of the self-organising architecture through the addition of the Error Generating Agent (EGA) the EGA contains a series of timers that issued commands to agents within the customer layer to trigger behaviours representing attack events. The EGA controlled when an attack event would begin and when it would conclude if the attack is not intended to endure for the length of the simulation.

5.5.1 Selecting Targets

Due to the nature of the initialisation phase, the EGA has no immediate knowledge of the communication structure of the agent population and therefore must be supplied with information from the Architect Agent. Once the initialisation stage is complete, the EGA would be supplied with lists of customer agents in contact with each of the aggregates – so that if the examined scenario required multiple customers to direct the attack towards a

particular target the EGA was able to determine which customers to select to initiate the attack.

Those names of the selected agents are then stored in an internal data structure such that the EGA is able to communicate with the attack population to terminate the attack when necessary. This also prevents subsequent attack events with the same simulation attempting to use the same attack population, removing the chance for duplication.

5.5.2 Performing Attacks

To perform the attack the customer agents were constructed with embedded behaviours which were either activated or deactivated by Boolean flags, these flags dictate whether or not the customer was launching an attack against its target. While the inclusion of these attack behaviours in each agent resulted in a large amount of unused agent code throughout the agent population it provided more control in the form of orchestrating an attack and shaping the nature of the attack. If each customer was only supplied with the behaviour required for each attack scenario, there would be small differences in how the agent population behaves at runtime due to the amount of virtual machine memory occupied by running the agents. Taking out unused code for one simulation, and then inserting it for another effectively changes the dynamic of the agent population between scenarios and ultimately makes the results less comparable. The overhead of running the agent platform with the increased code content and therefore the increased computational footprint did have an impact on the scalability of the simulation. The following diagram in Fig. 5.20 illustrates the interaction involved in launching an attack against a controller.

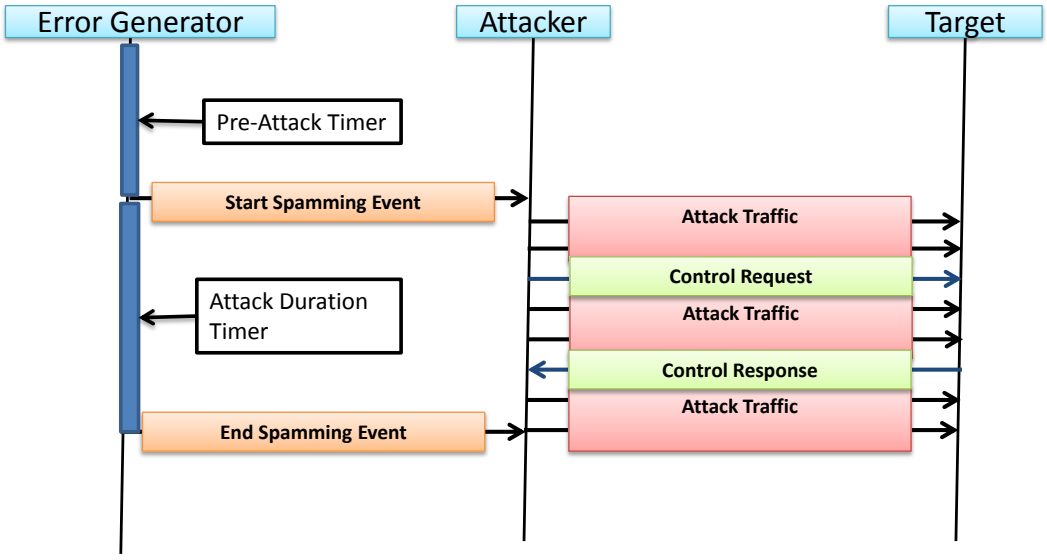


Fig. 5.20 – Launching an Attack

To activate attack behaviour, messages are transmitted from the EGA to the targeted population; the number of attackers is determined by the intent of the scenario. The distribution of attackers can be defined within the EGA without interfering with any configurations for the other agents in the population.

5.5.3 Timing Attacks

Part of the role of the EGA is to schedule the attack events in relation to the actions taking place within the core simulation, as the previous figure illustrates the attacks would be more effective if designed to coincide with key operations. In the simulation there are two stages where critical commands are sent from the controller to the customer population, these commands oversee the initial control commands and the removal of any controls imposed on the customer layer. The specifics of the attack methodology will be described in chapter 7, but the EGA is responsible for ensuring that the attack event is triggered during one of these key stages thus posing the risk of creating the largest disturbance. Multiple timers are employed by the EGA, the first of which determines the start of the first attack signal, other timers determine the length of the attack event and if necessary the start and end times of subsequent attack conditions.

5.6 SUMMARY

This chapter outlined each of the components involved in the development of the self-organising architecture. The system contained three stages of operation each informed by literature and described in the chapter in terms of the functions, communication requirements and their impact on the agent population. The initialisation stage is the first of those three stages, and involved forming connections between members of the customer layer – which included smart-meter based agents and small scale DG entities – and the aggregation layer. The second stage considered the use of performance monitoring techniques and examined the individual thresholds applied to the metrics embedded within the self-organising architecture. The performance monitoring stage involved the collection of error reports from members of the agent population along with the appropriate storage and processing approaches. A final stage considered the different architectural transition events the Architect agent had the ability to invoke in the event of a performance violation, each transition is described in terms of the functions required to perform the transition and the sequence of messages between entities.

In addition to the set of transitional mechanics, the processes for emulating attack events and the role of the Error Generation agent in the architecture. Attack behaviours were instantiated within each of the main agents involved with the self-organising architecture, and were activated by the error generation agent. The following chapter is focussed on the decision making engine which was responsible for converting the set of error information retrieved in during the performance monitoring stage into a transition event. This chapter discusses the different approaches to decision making and the mechanisms for evaluating the set of errors received by the architect agent.

Chapter 6: Decision Making Engine

6.1 INTRODUCTION

The decision making engine at the heart of the self-organising architecture was responsible for converting performance monitoring data retrieved from the agent population into transition events. Instead of a predetermined set of system reconfigurations the role of the decision making engine was to assess the conditions of the agent architecture through the provision of error reports such that an appropriate restructuring process could be applied.

Two implementations of the decision making engine were introduced initially centring on a decision tree mechanism breaking down the set of error responses into their individual sources. Each source equated a branch in the decision tree, where specific error context and location information resulted in a final decision on which of the architecture transition events to pursue. The second and favoured implementation replaced the decision making engine with a more sophisticated fuzzy based system – this alternate system involved computing a single variable for computational burden which accumulated data from each of the error formats involved with the communication aspect of the overall system. This decision making engine would then produce a recommendation as to the scale of the transition event, a recommendation which could either be accepted and implanted by the architect or over-ruled in the event of contradictory information.

The remainder of the chapter discusses the formulation of the fuzzy decision making system including the integration of the fuzzy membership functions into the java platform. Finally the chapter considers the processes involved in triggering a transition from the result of the decision making engine.

6.2 ERROR FILTRATION

Before a call to the decision making engine was triggered, a series of checks were performed on the list of error reports received from the agent population. These checks were a preventative measure to avoid transitional events being initiated on the basis of reports which may be out of date or that represented transient occurrences of a given error type and therefore prevented the overall decision process from being too sensitive.

As indicated in the previous chapter, all error reports for each of the performance metrics are transmitted to the architect agent, and are then stored in an overall error list data structure. The architect agent was responsible for processing this list of errors and sculpting the call to the decision making engine. In the presence of the initial decision tree approach,

this call took the form of determining the dominant error type in terms of its severity value. The list processing stage involved error filtration where each of the reports received by the architect agent was examined against a series of criteria before it was considered for use in calculating the overall error state of the architecture. In each case the error reports only existed for a finite length of time, and as discussed by the authors of [118] will expire and thus be no longer valid for consideration in terms of error analysis. A list of the set of filtration limits is presented in the following table

Table 6.1 – Performance Monitoring Thresholds

Metric	Threshold	Comment
Lifespan	30 seconds	The lifespan of an error determined the time after which the error is deemed to have expired. This was not the overall duration of the error state, but the time since the last occurrence of the error was reported. Therefore if no new reports of an error were received by the Architect, the record of the error was removed once the lifespan period had elapsed.
Severity Buffer	10%	A severity buffer prevented the decision making process from acting upon each event – so that excessive changes were not made.
Waiting Period	20s	Under the decision tree approach, a waiting period was instantiated to prevent the architect performing a transition on a transient event. This waiting period was then applied to the estimated duration of an event under the second decision iteration of the decision making engine after error rate of change was calculated
Standoff Period	Small – 120s Medium – 180s	Three standoff sizes were used for different magnitudes of self-organised response. A small change such as transferring customer connections had a smaller impact on the overall network and therefore brought with it a smaller waiting time. Larger reconfiguration events, such as

	Large – 300s	activating dormant agents or promoting an aggregation tier required a larger waiting period

6.3 DECISION TREE

The process of defining which course of action to take as a result of performance monitoring data was initially the jurisdiction of a decision tree approach. An analysis stage, processing the error report list determined which of the errors present was the most severe. Each of the potential error formats was represented by a branch in the decision tree, and these decision branches are documented as follows:

6.3.1 Control Errors:

A control error was a consequence of a voltage excursion lasting longer than the prescribed threshold. Rather than waiting for the completion of the control action, any customer agent affected transmitted an error report as soon as the voltage had remained outside limits for a period of time longer than the threshold value. Depending on the current level of control in place the architect would take steps to replace or relocate the control responsibilities. If the control was placed at observer – i.e. centralised control architecture was in use, the Architect can only move the control functions further down the network. Likewise if the control was placed at the lowest tier – with the customer agents, the Architect could only move to less decentralised control architecture. The only control architecture with replaceable controllers, in the event that a single agent with the control functions failed to achieve the desired performance was the aggregation control architecture. This was because the other agents in the MAS represent physical components and therefore could not be disconnected without losing data and potential controllability. Raising control level in one agent without mirroring that action in another would result in hybrid control architecture where a proportion of the customer population may send control requests to the generation agents, while others communicate with the aggregate layer. A hybrid structure would create complications when performing further transitions within the architecture. The decision tree branch for processing this error format is presented in Fig. 6.

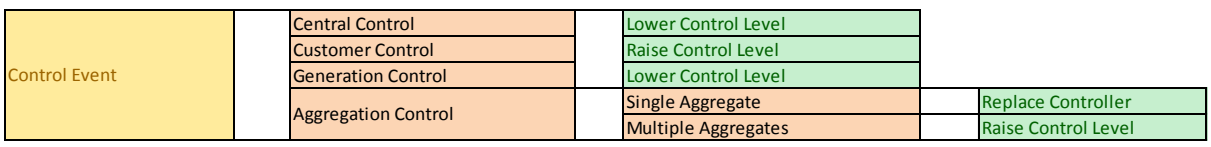


Fig. 6.1 – Decision Branch for Control Events

6.3.2 Data Flow Errors

In the same manner that control errors are declared, data flow errors were triggered when either incoming or outgoing data flow exceeded threshold values. Data flow errors were likely to be sourced from agents within the hierarchy responsible for handling larger volumes of information in the form of the aggregation layer agents and to a lesser extent the central observer. If the Observer was triggering data flow issues then it could be due to the number of aggregates transmitting customer data to it or if a member or members of the aggregation tier were launching an attack event. Therefore the solution was to add an aggregation tier to create a buffer between itself and the large quantities of information handled by the lower aggregation layer and potentially shielding itself from further attacks. As a secondary check the Architect also determines how many of the lower aggregates are producing data flow errors themselves, in order to decide which agents to promote to a new upper tier. Moving the aggregates with error reports places them in a less heavily loaded position within the hierarchy and therefore combats the data flow issues they are facing. Those with high outgoing data rates however were less likely to be considered for promotion, the decision tree branch processing data flow errors is presented in the following figure Fig. 6.2.

Data Event		Observer Affected		1 or no aggregates reporting errors		Promote one upper tier aggregate
				Multiple aggregates reporting errors		Promote mulitple upper aggregates
		Single Aggregate		Also congested/ slow response time		Transferring Connections
				Only showing Data Errors		Replace Agent
		Multiple Aggected		Dormants Available		Activate Dormants
				No Dormants Available		Promote Customer Agents

Fig. 6.2 – Decision Branch for Data Events

6.3.3 Congestion Errors

The congestion error metric was concerned with the number of messages that an agent was storing in its message queue, a larger message queue indicated a greater congestion problem. As was the case with the data flow error type, the congestion issue was primarily restricted to those agents responsible for interacting with a larger population and handling larger volumes of data. Furthermore because the two metrics are often closely related the decision branch for the congestion error was very similar to that of the data flow branch as illustrated in the following figure in Fig. 6.3.

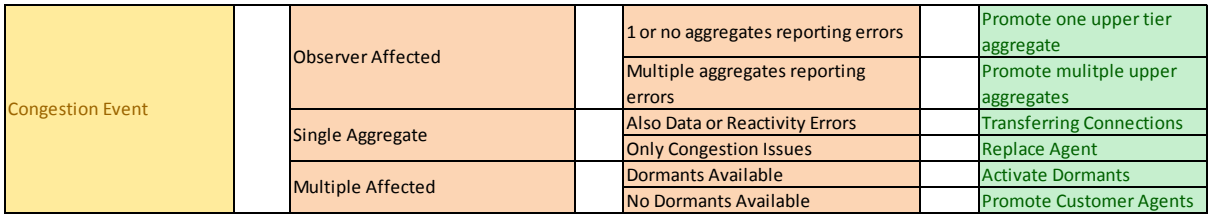


Fig. 6.3 – Decision Branch for Congestion Errors

6.3.4 Reactivity Errors

Reactivity errors were processed differently primarily because they are not observed and reported by the agent causing them Instead they were observed by agents that interact with the agent causing the problem. This allowed the architect to build a picture of the scope of the error by determining the percentage of customer agents reporting reactivity issues. Additionally the aggregation layer could report reactivity problems when communicating with the observer, and thus request a tiered architecture to alleviate some of the pressure on the observer agent. Therefore in terms of the decision branch as presented in Fig. 6.4, a check for impacts on the observer was preformed first before then assuming that the reactivity problem is occurring between the customer layer and the aggregation layer.

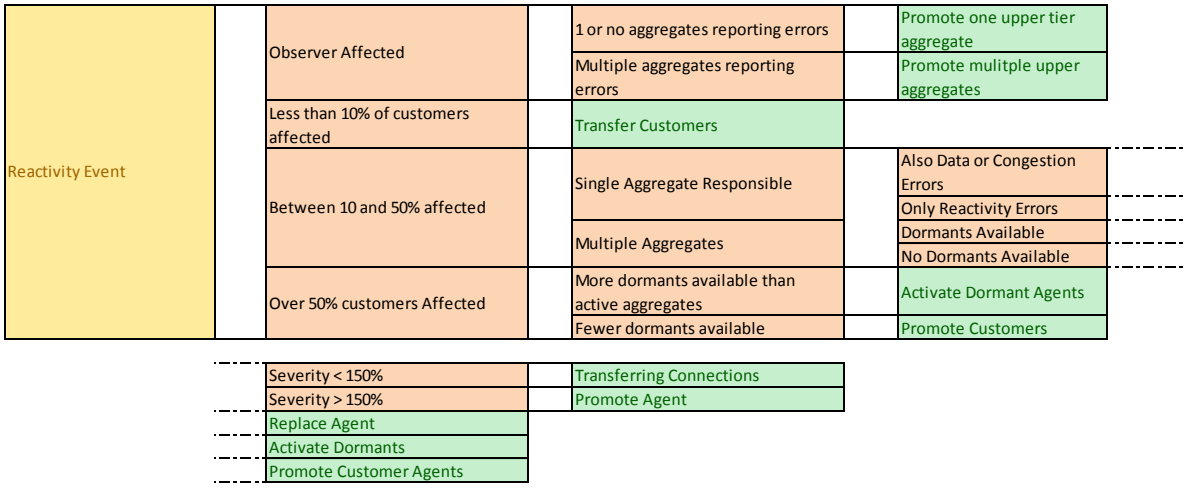


Fig. 6.4 – Decision Branch for Reactivity Errors

If only a small proportion of the customer population is affected then those customers could be relocated to alternative aggregates. If a larger population was experiencing difficulties then it became more relevant to examine the number of agents causing the reactivity problem. As an additional check, in the event of single aggregate causing problems, the magnitude of the severity is considered. If the severity was smaller than 150%, then it may be possible to solve the problem through rebalancing the communication load and transferring connections. If the severity was larger, then the aggregate agent could be moved to an upper tier where it would be exposed to less communication load. This would

be applied if the aggregate in question was experiencing muliple error types in addition to reactivity, as the performance metrics are often linked. However if the aggregate was only experiencing a reactivity problem it may be assumed that the agent is not functioning correctly and therefore can be replaced by subsitution.

6.3.5 Unresponsive Errors

The unresponsive error type was an extension of the reactivity error type in the sense that it refers to the interaction between agent pairs and the requirement of a recipient agent to reply with a confirmation message that it received the updated information. In the event that the recipient did not transmit this confirmation message the customer logs the message as missed, if ten messages are missed then the agent sends an unresponsive error alert. This was similar to the reactivity error type as the architect could determine the scope of the impact through the number of customers reporting the problem. A smaller error scope can be processed through relocating some of the connections to an aggregate with a smaller number of active connections. Whereas a larger error event would be processed through substituting or promoting low performing aggregates. If several aggregates were involved, a more widespread reorganising strategy was required on the basis of the number of customers affected. At an intermediate level either dormant agents would be promoted or the aggregation tier would be widened through promotion. At the most severe level the architect will select the solution which creates the largest aggregate population. The decision tree branch is presented in the following figure Fig. 6.5

Unresponsive Event		Single Aggregate Responsible		Less than 10% customers Affected		Transfer Connections		
				More than 10% affected		Multiple errors present		Promote Agent
						Unresponsive Only		Replace Agent
		Multiple Aggregates Responsible		Less than 10% customers Affected		Transfer Connections		
				Between 10 and 50% customers Affected		Dormants Available		Activate Dormants
						No Dormants Available		Promote Customer Agents
				Over 50% affected		More dormants available than active aggregates		Activate Dormant Agents
						Fewer dormants available		Promote Customers

Fig. 6.5 – Decision Branch for Unresponsive Errors

6.3.6 Underused Errors

The notion of monitoring agents for under usage was based on the ability to move specific agents into a dormant state if they running a small data flow measurement – as suggested by the authors of [106]. In the decision branch presented in Fig. 6.6, the response to each case of under-usage was dependent on the role of the agent submitting the error report. So each case for each of the agent types was handled in turn.

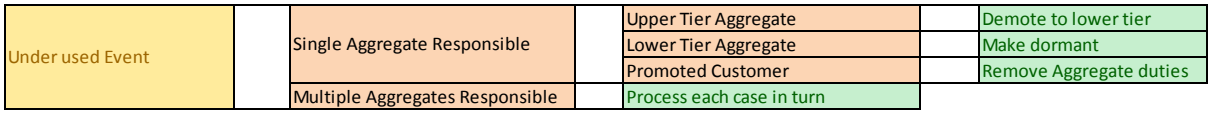


Fig. 6.6 – Decision Branch for Under Used Errors

An upper tier aggregate with low usage indicated that it would be better suited to being demoted to the lower aggregation layer as there wasn't the need for an upper tier, or a less populated upper tier. Also a customer agent which has been promoted into an aggregation role or substituted to take over aggregation responsibilities would monitor usage to determine if those additional responsibilities were still required. Finally an aggregate in the core aggregation tier reporting under usage may be made dormant and its connections distributed between the remainder of the aggregation tier.

6.3.7 Isolated Errors

The final error type is an isolated error, which is declared if a customer or generation agent cannot pair itself with an appropriate aggregation agent. This would occur during the initialisation phase if the customer had submitted requests to all available aggregates and been refused a connection slot. Isolated errors are also likely to occur if the network population changes over time, for example if an additional feeder was connected to the network and the customer population from the new feeder needed to interact with the MAS architecture the aggregate population may not have enough capacity to host the new connections. Therefore the architect may need to activate additional capacity to handle the connection demand. After first detecting the need for addition aggregates the architect can then forward subsequent messages to the new aggregate as it was unlikely that the new customers will have discovered the newly created aggregate and would still claim to be isolated. The decision branch responsible for processing isolated errors is presented in the following figure in Fig. 6.7

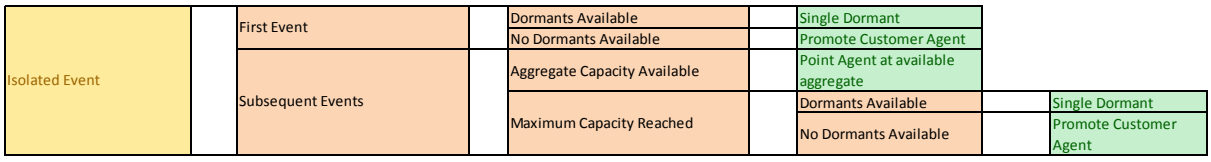


Fig. 6.7 – Decision Branch for Isolated Errors

6.4 DECISION TREE PERFORMANCE

The decision tree process needed to be evaluated once it had been installed within the architect agent to determine its effectiveness. A test configuration was devised consisting of 340 customer agents representing domestic smart-meters, 50% of the customer

population were declared controllable with a maximum reduction of 700W. Four active aggregates and two dormant aggregates formed the data collection tier; this tier was also responsible for performing control actions. The initialisation stage of the self-organising architecture defined that customers could select their own connection points, and therefore neighbouring customers could be connected to differing aggregation agents, this was true both in terms of controller and data collection connections. This process was developed to improve resilience within the architecture through control redundancy – agents tackling the same voltage problem could request assistance from alternate controllers. However those controllers could only request control responses from those agents affiliated with them during the initialisation stage. Each customer agent and generation agent was supplied with profile data which was compressed to cover the length of the simulation whilst presenting a voltage deviation which required a control response. Load flow calculations are completed on request from the gateway agent at the boundary of the JADE/Matlab interface (to be documented in chapter 7).

These initial results considered both a static and self-organising version of the architecture, and did not contain any attack vectors, only the voltage deviation was present

6.4.1 Static Architecture

To demonstrate the effectiveness of the self-organising architecture with the decision tree based decision making engine a base test was needed using a static version of the architecture. In the case of the static architecture, the initialisation phase locks connections between customers and the closest aggregate, performance monitoring components remained active but the decision making element of the process was disabled. Therefore the structure of the architecture remained the same throughout the simulation period, and any error conditions would not be mitigated by the actions of the architect agent. The following graph in Fig. 6.8 illustrates the errors received by the architect agent during conventional operation including a voltage excursion.

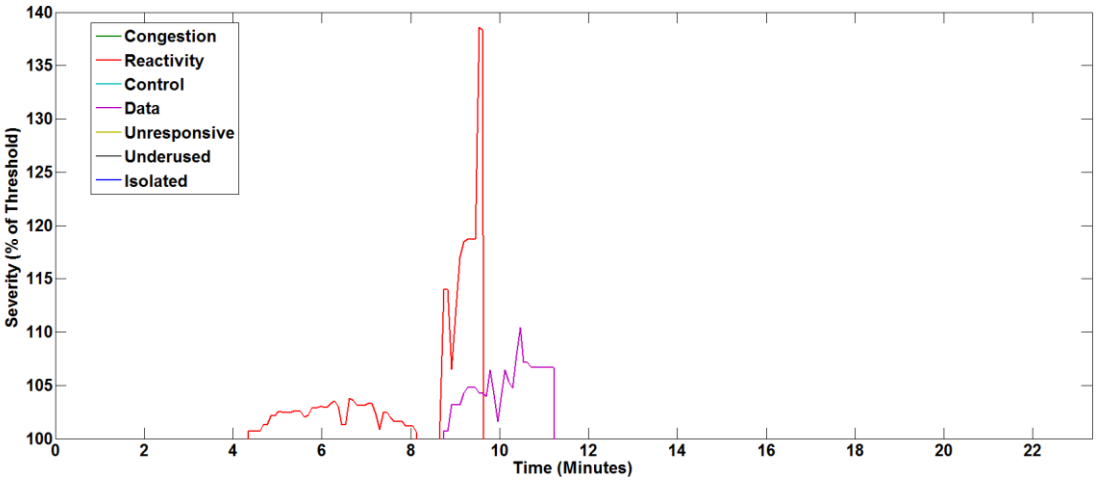


Fig. 6.8 – Error Severities Graph

The most dominant error type was a reactivity error event with a maximum severity of 138% - meaning that the largest reactibity event was 38% greater than the reactivity threshold. A second error event recorded was a data-flow error, however this error event did not exceed the 110% severity threshold value. In addition to recording the magnitudes of error severity, additional performance metrics were recorded during the course of a simulation, these performnce metrics included the length of the voltage devation, the amount of time customers were placed under demand restrictions the details of the voltage profiles taken from the end of the feeder. A further data point recorded was the demand losses as a result of customers being placed under restriction. To determine the amount of consumption lost through the control method, each customer recorded its raw demand as per its load profile in addition to the actual demand including control restrictions. These figures were compared for each sample in the output file to determine the consumption difference in kWh for each step as illustrated by the following equations (1-4) supported by the notation described in Table 6.2.

(1) $$t_{dist} = \left(t_{(n+1)} - t_n\right) \times 2.77778 \times 10^{-7}$$

(2) $$P_{kWh} = \frac{(P_{Watts} \times t_{dist})}{1000}$$

(3) $$P_{total} = \sum_{S_n}^{S_1} P_{kWh}$$

(4) $$P_{losses} = P_{total} - P_{restricted}$$

Table 6.2 – Consumption calculations notation table

Notation	Description

t	Timestamp of a given sample
t_{dist}	The time between two samples in hours
P_W	The real power recorded for a given sample
P_{kWh}	Real power consumption for the lifetime of a given sample
S_n	Sample number within the output file
P_{total}	Total consumption without restriction during a simulation
$P_{restricted}$	Total consumption with restriction during a simulation
P_{losses}	Total consumption losses due to customer restrictions

In the first equation the length of time each sample was valid for was calculated and the time between the current sample and the next one was converted from milliseconds to hours to complete the kWh conversion. The conversion presented in equation (2) was applied to each sample in the output file for the output field referring to raw unrestricted customer demand. Therefore a total consumption figure could be attained representing consumption if no restrictions were applied as documented in equation (3). This process was repeated for the output field referring to actual demand including control restrictions to give two consumption totals. From this information the losses could be determined as presented in equation (4).

The results of the static architrecture test example containing the affoermenttioned metrics is illustrated in the following table in Table 6.3.

Table 6.3 – Performance Results

Error Performance	Congestion	Reactivity	Control	Data	Unresponsive	Under-Used	Isolated
Maximum	0	138.57	0	110.4	0	0	0
Average	0	105.48	0	105.13	0	0	0
Voltage Performance	Min Voltage	Max Voltage	Average Voltage	Total Time	Single Event Maximum		
Feeder 1	0.96073	0.92644	0.94445	371.42	328.69		
Feeder 2	0.96069	0.92631	0.94438	383.8	331.93		
Feeder 3	0.96033	0.92652	0.94412	384.01	329.69		
Feeder 4	0.96004	0.92645	0.94417	405.06	330.04		
Control Performance	Actual Demand (kWh)	Without Restriction (kWh)	Total Restricted (kWh)	Average Restricted (kWh)	Total Restricted (%)	Max Total Restriction Time (s)	Max Single Restriction Event (s)
Feeder 1	31.459	35.69	4.232	0.101	11.86	706.79	706.79
Feeder 2	31.451	35.595	4.145	0.096	11.64	703.3	703.3
Feeder 3	31.303	35.601	4.297	0.102	12.07	704.13	704.13
Feeder 4	31.333	35.715	4.381	0.102	12.27	704.39	704.39
Overall	125.546	142.601	17.055	0.10025	11.96	704.6525	704.6525

In the results derived from the example of operating under a static architecture customer agents were restricted for 704 seconds on average as a result of controls applied to correct the voltage deviation. This restriction period caused a total consumption loss of 17kWh, averaging at 0.1kWh per customer accounting for 12% of total demand during the 704 second window. This information served as a further point of comparison between the static and self-organising architectures in determining where the performance gains could be achieved.

6.4.2 Self-Organised Architecture

In the second test case, the self-organised functions were activated, allowing the architect agent to respond to the error reports delivered by the agent population. Because there were no scheduled attack or failure events the level of restructuring required was reasonably small, and therefore only a single decision was made during the simulation window. The following graph in Fig. 6.9 presents the error severities received by the architect. The figure also illustrates that only a reactivity error exceeded the 110% buffer window and therefore triggered a transition event.

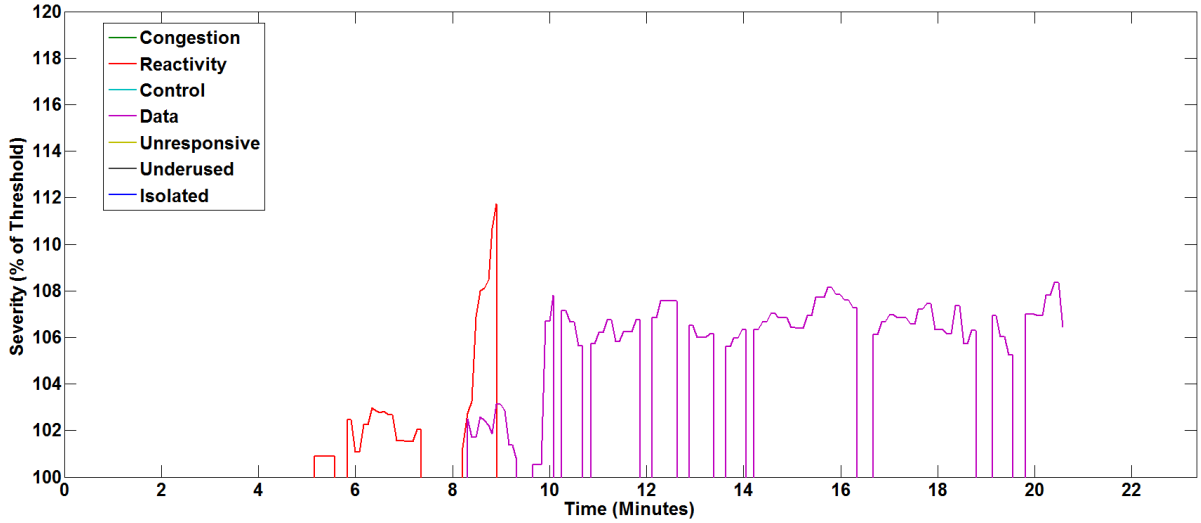


Fig. 6.9 – Severities Graph

The transition event was a rebalancing event, where a number of customer connections were transferred from one aggregate to another. In the example, customers from aggregates three and four are moved across to aggregate one – customers reporting the reactivity issue were targeted and relocated to an aggregate with available connection capacity. This helped distribute the connections more evenly in the event that the initialisation stage created an uneven distribution of communicative load between the different aggregation points. The following figure presented in Fig. 6.10 illustrates the changes in data flow in at members

of the aggregation layer during the simulation and illustrates the impact of the rebalancing transition.

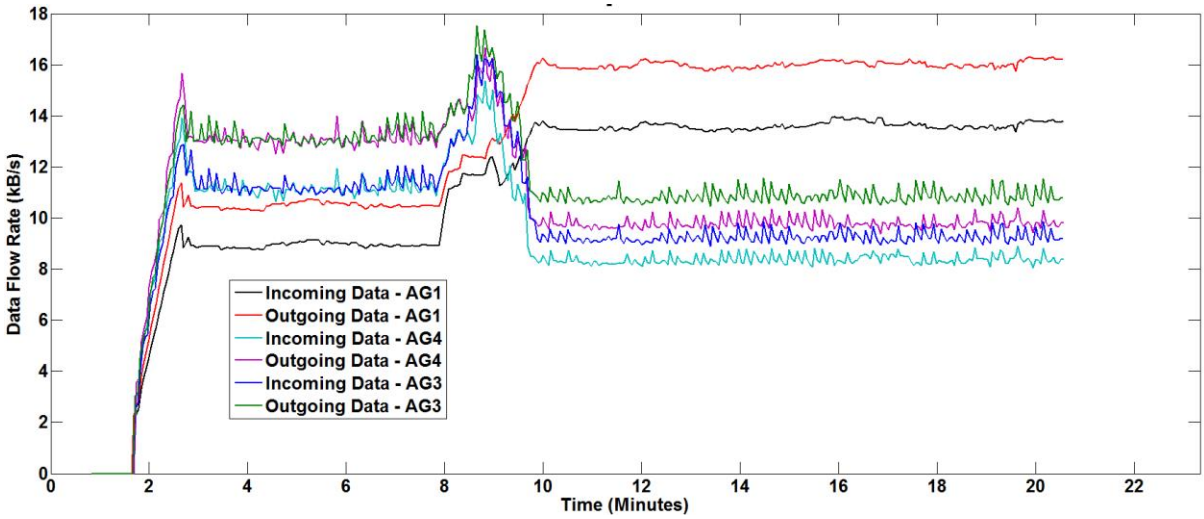


Fig. 6.10 – Data Flow at the Aggregation Layer

Aggregates three and four observed a rise in communicative load nine minutes into the simulation; this corresponded with the series of requests for control in correcting the voltage deviation, which suggested that they were not balanced correctly during initialisation. Therefore the architect elected to perform a rebalancing action and move a set of connections to the first aggregate; this is presented in the figure through a reduction in both incoming and outgoing data flow at aggregates three and four. Aggregate one however observed an increase in data flow, this did not translate into a reactivity issue in terms of overall error severity, furthermore the data flow increase remained within the 110% error severity threshold.

The additional performance metrics retrieved from the simulation are presented in the following table in Table 6.4. The values in the table represent the difference in percentage terms between the tests performed on the static and self-organising architectures. A negative value determines that the self-organising architecture delivered improved performance, while a positive number presents a performance loss.

Table 6.4 – Comparative Performance Metrics

Error Performance	Congestion	Reactivity	Control	Data	Unresponsive	Under-Used	Isolated
Maximum	0.00	-24.03	0.00	-1.87	0.00	0.00	0.00
Average	0.00	-2.28	0.00	0.90	0.00	0.00	0.00
Voltage Performance	Max Voltage (feeder end)	Min Voltage (feeder end)	Average Voltage	Total Time	Single Event Maximum		
Feeder 1	0.02	-0.001	0.07	-5.63	-5.19		
Feeder 2	0.03	-0.001	0.08	-12.05	-6.23		
Feeder 3	0.08	-0.004	0.07	-1.66	1.35		
Feeder 4	0.07	0.003	0.07	-8.58	0.71		
Control Performance	Actual Demand (kWh)	Without Restriction (kWh)	Total Restricted (kWh)	Average Restricted (kWh)	Total Restricted (%)	Max Total Restriction Time (s)	Max Single Restriction Event (s)
Feeder 1	3.33	2.56	-3.70	-4.12	-6.46	6.96	6.96
Feeder 2	3.19	2.47	-3.32	-3.23	-5.91	7.32	7.32
Feeder 3	3.45	2.61	-4.02	-4.08	-6.81	7.05	7.05
Feeder 4	3.40	2.66	-2.99	-3.03	-5.87	7.05	7.05
Overall	3.34	2.57	-3.50	-3.62	-6.26	7.10	7.10

The results demonstrate that there was a notable improvement in several metrics, considering the test was performed on a standard simulation without an attack or failure event in place. The maximum magnitude of a reactivity error event was reduced by 24%, which was corrected by performing a rebalancing action. From an electrical point of view, the deviation length was reduced by up to 12%, however this did result in longer restriction times as the controls were applied sooner as a result of the improve system reactivity.

6.5 LIMITATIONS AND CONCLUSIONS

The decision tree method had the capability to process the list of errors and ascertain which of the error reports was the most severe and therefore make a decision based on this information. Furthermore when executed the decision making process was able to select a transition event and execute it, which in turn was able to reduce the severity of the selected error condition and lead to improvements in the performance of the electrical network. However other tests and investigations indicated that the decision tree was subject to several limitations in terms of how it was implemented, its flexibility and the manner in which inputs were processed.

The first of these limitations concerned the nature of the decision tree approach, each of the rules needed to be explicitly designed and coded into the architect agent and therefore if new performance metrics were to be included the entire rule base would need to be redesigned. This would also lead to large unmaintainable rule-sets if multiple control objectives and conditions needed to be satisfied. A second weakness was that the decision

tree mechanism relied on a series of crisp binary decisions, whether each decision statement was true or false which did not suitable model the nature of the problem. Several sources of uncertainty were present within the threshold selection stage, error severities and locations – therefore employing a decision system which made certain decisions without considering degrees of truth was not necessarily the correct format. Finally the decision tree approach required the separation of error formats, and didn't have the same focus on determining the overall state of health of the architecture. For example if the most severe error is a reactivity problem traced recorded by several customers, a rebalancing action may be triggered, however other issues within the architecture may suggest that a larger scale transition is warranted. The process of fine tuning the decision tree would involve the resulting of the rule set and further raise the issue of maintaining an extensive set of decision branches.

Overall the decision tree was able to perform error processing, and select the most prominent error format present within the architecture. This error format formed the input to the decision tree allowing the Architect agent to select an appropriate transition event and execute it. This process was able to reduce the severity of errors in the system and ultimately create improvements in deviation duration and customer losses. These improvements were present in the absence of any attack event therefore it was reasonable to assume that in instances where the net performance of the architecture was reduced the potential for performance gains were greater during an attack event. Indicating that the transitions triggered by the decision making engine were effective and achieved the objectives required of them. However the outlined limitations of the decision making engine itself indicated that an alternative approach was needed in selecting the appropriate transitional action. This alternative approach needed to be focussed on flexibility and performing transition decision making in an environment which was rich in uncertainties.

6.6 A FUZZY BASED DECISION MAKING ENGINE

In recognition of the limitations of the decision tree mechanism a replacement system was based on a fuzzy decision making method. The alternative solution involved combining a subset of the performance metrics into a single computational burden value representing the overall error state of the architecture. The computational burden figure and its rate of change were used as input membership functions to the decision making engine. In terms

of output functions, the decision making engine recommends a transition stage referring to the following:

- Stage 1: Rebalancing Action
- Stage 2: Substitution
- Stage 3: Activating Dormant Agents
- Stage 4: Tiered Aggregate promotion

A higher burden would be associated with a higher stage transition, which involves a more drastic reconfiguration event. To adopt principles from the decision tree method, the architect does not automatically have to accept the recommendation from the fuzzy decision making engine because it is aware of other criteria which would influence selecting a transition. The other criteria included the location of the errors detected, the availability of resources for promotion or dormant agent activation and the distribution of errors. From this additional data the architect had the power to override the recommendation through truncated decision tree mechanisms.

6.6.1 Computational Burden Components

The two inputs into the fuzzy model were the computational burden and the rate of change of that burden. This involved creating a metric which alluded to the severity and scale of a series of error types referring to the communication and computational processing elements of the agent community. The computational burden value aimed to produce a figure describing how much computational stress the architecture was under at any given time, this figure was an amalgamation of the following performance metrics as recorded by the agent population. The following metrics are used to build the computational burden figure.

Congestion – The number of messages stored in messages queues at an agent waiting to be processed

Data Flow – Incoming and outgoing data flows are recorded at each point in the network and are subject to thresholds describing the maximum amount of throughout that can be sustained without increasing the risk of data losses when considering a ZigBee communication platform

Reactivity – Response times between agent pairs – the time taken between a transmitted update and the corresponding reply from the target. This time includes the amount of time

the message has spent in the message queue of the recipient agent and the time it takes for the recipient the process the message and delivers the response.

Execution Time – An internal form of reactivity which measures the time from the point a message has been received to the point where the recipient decides which response to select and perform that selection.

Unresponsiveness – Acts as a more extreme form of reactivity measurement, in the sense that if an agent does not receive a message within a specified timeout window from the agent it is communicating with it records that agent as being unresponsive. Multiple successive missed messages trigger an error alert which is sent to the Architect.

Each of these error types was accompanied a magnitude, a threshold and timing information – which is then stored in the architect's 'Error Base'. The error base was a set of linked objects which document and store lists of all error reports fitting each of the listed error types such that each type could be processed separately. The use of the knowledge base provides greater flexibility in the event that other performance metrics were to be added, each subsequent metric would be assigned an instance of an error type object which hosts all reports for that given type.

6.6.2 Computing a Computational Burden Indicator

After filtering the set of error reports, each of the reports pertaining to the error types outlined in the previous section was combined to create an overall burden value. The following equation in (5) presents the calculation required to compute the figure for the computational burden indicator.

(5) $$burden = \frac{\sum_{et_n}^{et_1}\left(\frac{\sum_{c_n}^{c_1} customer\ error\ severity}{number\ of\ customers} + \frac{\sum_{ag_n}^{ag_1} aggregate\ error\ severity}{number\ of\ aggregates}\right)}{number\ of\ contributing\ error\ types}$$

The first step involved calculating the average severity for each of the contributory error types with respect to the tier in which those errors are being detected. For example a series of reactivity errors recorded at the customer layer will be averaged with respect to the number of customers, whereas a series of errors of the same type within the aggregation tier was averaged against the number of aggregate agents. The averages for each agent tier are added to form an architecture wide value for each of the error types. The final

computational burden figure is derived from the architecture wide value divided by the number of contributory error types. A second component of the computational burden assessment was the rate of change of the figure. The overall burden data was monitored using the same techniques each of the agents within the core architecture population used to perform the performance monitoring duties. A performance monitor object in the Architect agent is supplied with the computational burden figure after each calculation. The performance monitor retains a list of recent calculation results to compute a moving average, changes in subsequent average calculations determined whether the computational burden was increasing or declining. Timestamps between average calculations were then used to calculate the rate of which the increases or decreases were taking place.

Because of the number of metrics involved in the calculation of the burden indicator, each of which was initially recorded with differing units – congestion took the form of a number of messages, whereas reactivity was measured in second – the final value for the burden indicator was declared a dimensionless quantity. A low, medium or high level of computational burden was defined by the indicator values defined in the following membership functions. Furthermore the uncertainty involved with an indicator rather than a crisp burden measurement increased the relevance for defining a fuzzy based decision making engine.

6.6.3 Membership functions

Following the calculation of a computational burden and the rate of change of burden, the next stage in determining which transitional event was required was to pass the two values to the fuzzy decision engine. The decision tree approach didn't apply a transitional event if the severity of the largest error was below 110%, thus avoiding the architect reacting to an event which posed little thread to overall performance. With this decision making engine the threshold limit with regards to action is set with a computational burden of 5, if the burden exceeded this base value the fuzzy decision making engine was triggered. Two input membership functions processed the two input parameters and are presented in the following figure in Fig. 6.11.

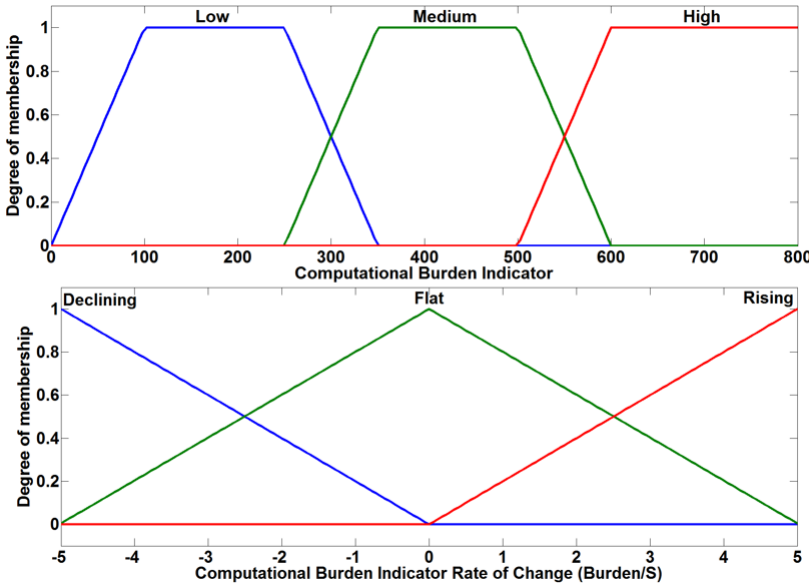


Fig. 6.11 – Input Membership Functions

The first input was focused on the magnitude of the computational burden categorising the burden level, three stages were defined within the membership function for low, medium and high computational burden states. The low membership function covered the smallest area because it represents a smaller amount of potential cases and was associated with fewer architectural transition mechanisms. The high membership function extended beyond the initial range presented in the figure as a result of there being no effective upper limit to computational burden. Therefore in order for the recommendation service to be able to deliver a response in those more extreme cases the upper bound of membership function needed to take this into account. There was a degree of overlap between the functions because the computational burden is an amalgamation of a number of performance metrics, it created uncertainty as to the actual severity of the error state. Furthermore setting the thresholds for some of the performance metrics relied on a series of tests under the presence of normal and abnormal communication load. These exercises indicated that depending on the results of the initialisation phase the relative communication load varied across different points in the aggregation tier. Therefore uncertainty is also present in the individual performance metrics based on difference between the normal and abnormal communication loads generated during an attack event. As a result of the uncertainties present in the system there was no definitive point whereby the burden transition from a low to medium or high state and the intermediate areas of the membership functions represented instances where two burden conditions were partially true.

The second function was responsible for processing the rate of change of the computational burden, as with the first function, the rate of change input was composed of three membership functions. Anything less than zero indicated that the error state was declining and therefore was less likely to require a transition event, whereas any value greater than 0 indicated that the situation was escalating. The rate of change was only ever truly flat when the rate of change was zero, therefore the degree of truth to which the flat membership function was satisfied decayed each side of the centre using a triangle function rather than a trapezoidal one.

The output membership function is presented in Fig. 6.12 which converted the fuzzy result into classifications for the four architecture transition stages; each of the transitional stages was represented by a symmetrical trapezoidal function with a degree of overlap.

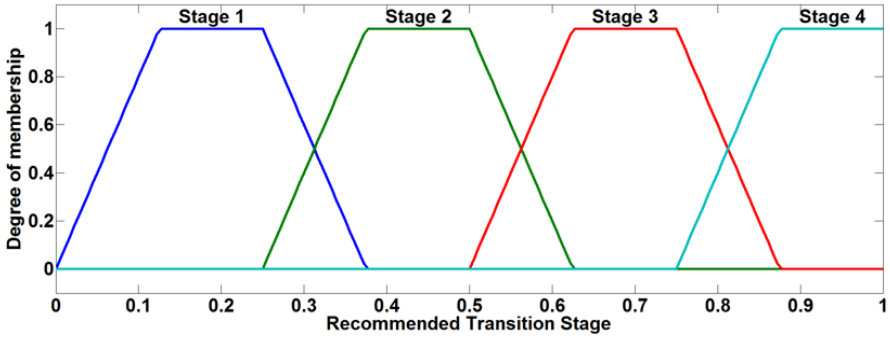


Fig. 6.12 – Decision Output Membership Function

These overlaps indicate the level of uncertainty in each of the potential transition stages indicating that certain scenarios may not present with a clear recommendation. In those instances the architect would then consider additional information pertaining to the overall error state in terms of the location and distribution of the error reports and the number of available dormant agents which could be used for the purposes of increasing aggregation capacity. Even in cases where the fuzzy system responds with a definitive transition recommendation the architect was required to perform a validity check to ensure that change which was recommended was feasible and could be implemented. If this was not the case the architect had the authority to over-ride the decision and apply a similar transition which adhered to the nature of the problem and the available resources.

6.6.4 Defuzzification and Rule Processing

With the input and output functions defined the final stage of configuring the fuzzy decision making engine was to define a rule set converting the inputs into a recommendation value which could be interpreted by the architect.

The table presented in Table 6.5 presents the rule base which was used to process the input membership functions and derive a transition recommendation. If the rate of computational burden was declining to such a degree that is it estimated that the overall error state will only persist for a short period of time the fuzzy recommendation function was not triggered. Only if the error state was estimated for a prolonged period of time would a recommendation be required. Additionally if a low computational burden is detected to be declining, no action will be taken by the architect as the condition is improving without the need for intervention. Larger incidents reporting with a declining burden were responded to as even in a declining state the error state would remain within the architecture longer than the threshold time.

Table 6.5 – Fuzzy Decision Making Rule Base

Computational Burden	Burden Rate of Change	Transition Stage
Low	Flat	Stage 1
Low	Rising	Stage 2
Medium	Declining	Stage 1
Medium	Flat	Stage 2
Medium	Rising	Stage 3
High	Declining	Stage 2
High	Flat	Stage 3
High	Rising	Stage 4

The overall ruleset indicated that the rate of change of the computational burden acted as a modifier with respect to the severity of the architect's response, an escalating condition was treated with a more severe action by the architect. In the same manner, a situation which was easing but still remained an ongoing problem was processed through the application of a less severe transition event. The final aspect of defuzzification process was the selection of the defuzzification method, all designs options were examined through the Matlab fuzzy toolbox and this process indicated that the defuzzification method could affect a small influence on the shape of the rule surface. This similarity between defuzzification methods has also been noted by the authors of [142] whereby centre of gravity, bisector and Mean of Maximum have all produced similar results. Therefore the

selected defuzzification method was centre of gravity and formed the surface presented in in Fig. 6.13.

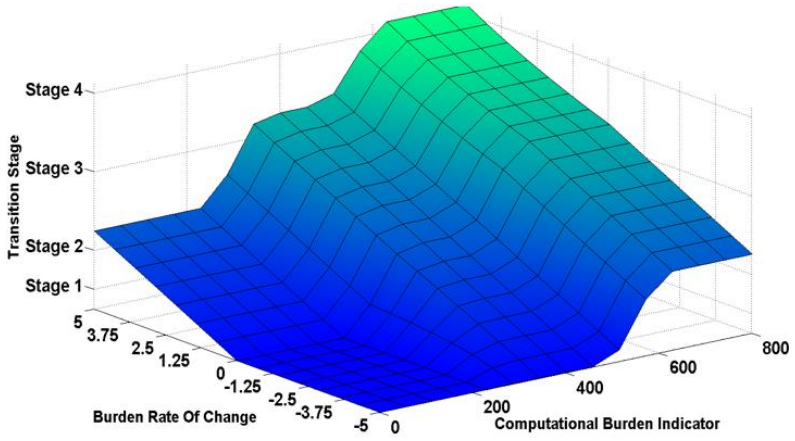


Fig. 6.13 – Fuzzy Rule Surface

6.7 JAVA IMPLEMENTATION

The initial fuzzy system design and evaluation was conducted within the Matlab fuzzy toolbox and integration with the agents involved in the self-organising architecture was conducted through a command line script triggering Matlab and the relevant function. However this process experienced several delays when initiating a new instance of Matlab to perform the function and to retrieve the result. Therefore an alternative implementation was introduced to alleviate these problems and increase the rate at which the architect agent could request a transition recommendation. The solution was to use a java library called jFuzzyLogic which could be accessed within the agent rather than making external function calls to Matlab. To utilise the library the membership functions and the rule list needed to be translated into a format recognisable by the library – which took the shape of a Fuzzy Control Language (FCL) file. The first element of this control language file was to define the input and output variables, as presented in Fig. 6.14; each of the components presented in this chapter was initialised in the source file.

```
VAR_INPUT
      burden: REAL;
      roc: REAL;
END_VAR

VAR_OUTPUT
      decision: REAL;
END_VAR
```

Fig. 6.14 – Defining Fuzzy Variables

Once the set of variables is declared the next stage defined the shape and scope of the membership functions for each of the variables. This segment of the FCL source file is illustrated in Fig. 6.15

```
FUZZIFY burden
      TERM low: = trape 0 75 175 250;
      TERM medium: = trape 175 250 500 575;
      TERM high: = trape 500 575 1000 1075;
END_FUZZIFY

FUZZIFY roc
      TERM declining: = trian -5 -2 0;
      TERM flat: = trian -2 0 2;
      TERM rising: = trian 0 2 5;
END_FUZZIFY

DEFUZZIFY decision
      TERM Stage1:= trape 0 0.125 0.25 0.375;
      TERM Stage2:= trape 0.25 0.375 0.5 0.625;
      TERM Stage3:= trape 0.5 0.625 0.75 0.875;
      TERM Stage4:= trape 0.75 0.875 1 1.125;
      METHOD : COG;
      DEFAULT: = 0;
END_DEFUZZIFY
```

Fig. 6.15 – Defining Membership Functions within the FCL File

The final stage of the FCL declaration file considers the rule set which in turn forms the rule surface, an identical set of configuration parameters were retained from the design of the system within the Matlab tool box and the eight rules defined in Table 6.5 were then translated into the FCL format. The translated rule base is presented in the following figure in Fig. 6.16.

```
RULE 1 : IF burden IS low AND roc IS flat THEN decision IS Stage1;
RULE 2 : IF burden IS low AND roc IS rising THEN decision IS Stage2;
RULE 3 : IF burden IS medium AND roc IS declining THEN decision IS
Stage1;
RULE 4 : IF burden IS medium AND roc IS flat THEN decision IS Stage2;
RULE 5 : IF burden IS medium AND roc IS rising THEN decision IS Stage3;
RULE 6 : IF burden IS high AND roc IS declining THEN decision IS Stage2;
RULE 7 : IF burden IS high AND roc IS flat THEN decision IS Stage3;
RULE 8 : IF burden IS high AND roc IS rising THEN decision IS Stage4;
```

Fig. 6.16 – Defining the Rule Set within the FCL File

The fuzzy library was imported by the architect agent only and was initialised at start-up by accessing the FCL file. When the architect agent required a transition recommendation

it supplied the fuzzy function with burden and rate of change data, and requested the evaluated result.

6.8 TRIGGERING A TRANSITION

With the output value retrieved from the fuzzy recommendation the architect then had to translate this into a reconfiguration action to apply it to the agent population. The output membership function defined which stage of transition is recommended for the given error state. Before accepting a recommendation the architect had to assess the feasibility of the recommended transition based on the current agent population status. For each of the transition options the architect had a series of checks to perform before deciding to perform the action suggested by the decision making engine. This incorporated elements from the initially formulated decision tree concept, and prevented the architect attempting transitions which would either not help the architecture or would not be possible based on resource availability.

6.8.1 Rebalancing

If the fuzzy system recommended performing a rebalancing action the architect checks to ensure such a change would have a positive influence. If the architect discovers that the agent which was the source of the computational burden error was not one which is connected to a large number of customer agents it concludes that rebalancing would be ineffective and would suggest a substitution instead. Likewise if the architect discovers that while the overall burden is relatively low, but sourced to a wide number of locations it would determine that a more reasonable transition would be to offer aggregation support as the overall agent population was under strain.

6.8.2 Substitution

An alternative recommendation would be to apply a substitution. The architect would look at location information for the set of error reports stored in the report list to determine of one aggregate was responsible for the errors. If so then the architect would carry out the desired transition as recommended, however if the error locations are more disparate then a substitution would not be as effective. Therefore a localised transition may not be the most appropriate, and a smaller scale version of a higher transition stage is applied, in the form of adding a single aggregation agent to the population.

6.8.3 Activating Dormant Agents

When a stage three transition was recommended the architect agent had fewer checks to perform. As this was a higher level transition event there were fewer transition options in the event that the initial recommendation cannot but fulfilled. The limitation in this case was the availability of dormant agents – when the agent population was initially launched only a small quantity of dormant aggregates was present. If these agents were activated, the architect cannot elect to perform another activation based transition as there were no more remaining dormant agents. Instead the architect over-rides the recommendation and proceeds to perform a single tier promotion transition. In this transition customer agents were promoted into aggregation roles within a single aggregation tier – one customer per existing active aggregate.

6.9 SUMMARY

This chapter documented the development stages of the decision making engine responsible for the delivery of architecture transition events after reviewing performance monitoring information retrieved from the agent population. Initially discussing processes surrounding error filtration such that only valid reports are considered for analysis, and that the architect does not respond to short term events or those with a trivial impact on the performance metrics. A preliminary decision tree method was outlined, discussing the responses to each of the error formats and demonstrating its effectiveness when presented with an attack format in a sample exercise. The overall performance and potential limitations of the decision tree approach resulted in the development of an alternative mechanism for translating performance data into transitional actions. This alternative mechanism involved the employment of a fuzzy based recommendation system which required the amalgamation of the performance metrics into a single variable. Therefore a value for computational burden was devised in addition to a process for computing the value from the supporting performance figures. This variable and its rate of change were used as input parameters to the decision making engine which returned a recommended transition stage from, the architect could either accept this result or override the decision based on additional information,

The following chapter considers the processes involved in evaluating the self-organising architecture presented in the previous chapter and the decision making engine documented in this one.

Chapter 7: Performance Evaluation Framework

7.1 INTRODUCTION

To evaluate the effectiveness of the self-organising architecture an evaluation framework was required, a framework defining an electrical network, an agent population and a simulation scenario. The goal of the platform was to demonstrate that the transitions and overall actions of the self-organising architecture were capable of having a positive impact on the performance of both the electrical layer and the communication layer.

The tests performed in the process of evaluating the self-organising performance differed from those presented in chapter 3 because of the nature of the evaluation objectives and the agent architecture involved. The previous set of tests was aimed at determining the potential for self-organisation through investigating a series of static architectures to extract performance information. Those results indicated that depending on the performance metric under evaluation differing architecture designs proved to be advantageous, and no single design was continually out-performing the others. Consequently that indicated there was scope for the introduction of a self-organising architecture. The evaluation processes documented in this chapter were aimed at investigating the effectiveness of a self-organising architecture which was developed in response to the results previously attained.

The evaluation platform described in this chapter considered two network events which required intervention from the agents within the architecture. The first of the events was the voltage deviation, a deviation which required the attention of the aggregate level controllers and the load shedding responses from the customer layer. Secondly the architecture was to be attacked by a low-rate denial-of-service attack; this cyber-threat was aimed at disrupting the actions taken during the control phase and therefore interrupting the dissemination of control signals.

In this chapter, the configurations for the components involved within the performance evaluation framework are introduced, including the structure and properties of the electrical network and the agent architecture. Furthermore the control problem and attack methods are also documented, concluding with the implementation of the environment within which the evaluations were performed. Finally the evaluation criteria are illustrated with respect to the electrical and communication layer performance.

7.2 TEST NETWORK CONFIGURATION

The core electrical network used throughout the evaluation process was derived from the network implemented for the initial series of static architecture tests; a radial configuration was developed as illustrated in Fig. 7.

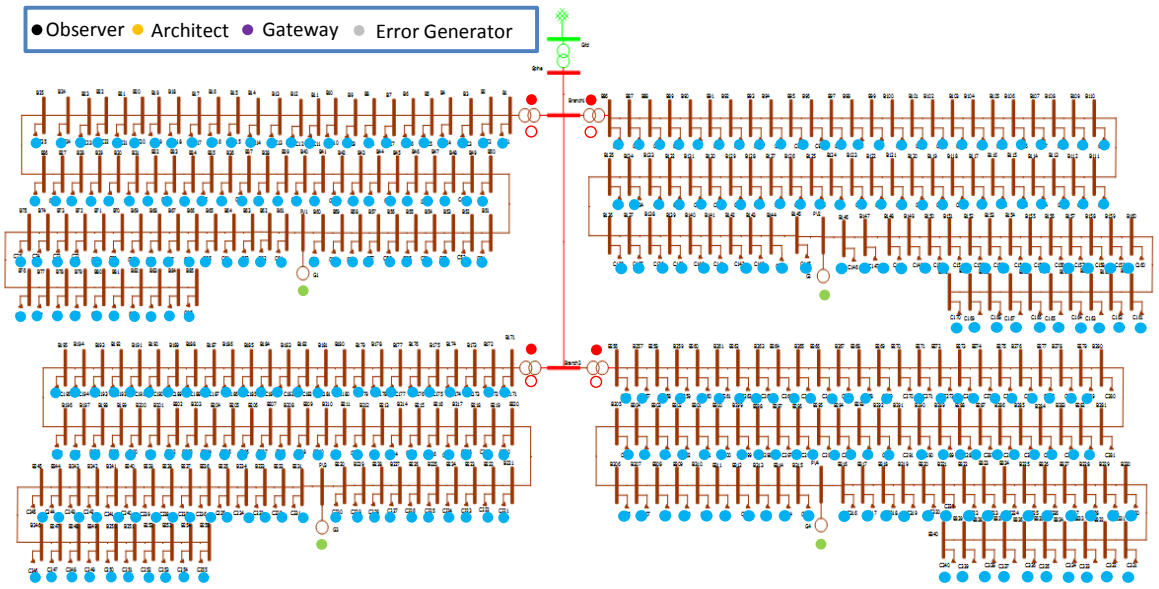


Fig. 7.1 – Network and Agent Topology Diagram

As illustrated in the diagram, the electrical network contained 340 customers, distributed across four radial feeders. Three voltage levels are presented within the network, starting with a 33kV grid connection, which was connected to an 11kV central network spine. The four feeders which hosted the customer connections were at the 400V voltage level, 85 customers were connected to each of those feeders. The customer agents were supplied with a demand profile which was sourced from the same data set as used in the first round of architecture evaluations. A modification was made to the source profiles in order to produce a voltage deviation with a significant enough duration to adequately examine the potential value of the self-organising architecture. This modification involved extending the period of high demand to create a longer term under-voltage situation.

Additional one PV installation rated at 10kW, and supplied with an appropriate generation profile was inserted after the 60th customer on each feeder. Also presented within the figure is the association between the components in the network and their agents, therefore illustrating the placement of the individual agents in respect of the electrical configuration.

The aggregation agents were placed at the head of each of the feeders, in addition to the four active aggregates; four dormant agents were also added to the initial starting configuration.

Beyond the agents which represented physical components or aggregation points in the network, further agents were located centrally and therefore were not associated with a specific component. These agents were the Overserved, Architect, Error Generator and Gateway agent – the latter agent being a function of the overall test environment discussed later in the chapter.

7.3 AGENT ARCHITECTURE CONFIGURATION

The previous subsection defined the structure of the physical network and discussed the location of the agents, where each of the customer smart-meters and generation components was represented by an agent. Additional agents were also present and were either located centrally, representing a cloud implementation or a central control room. However this only described the location of the agents involved not the communication structure and architecture. The interaction between the agents followed a typical architecture as presented in Fig. 7.2, this configuration served as the starting architecture for the self-organising architecture but also the as the static architecture.

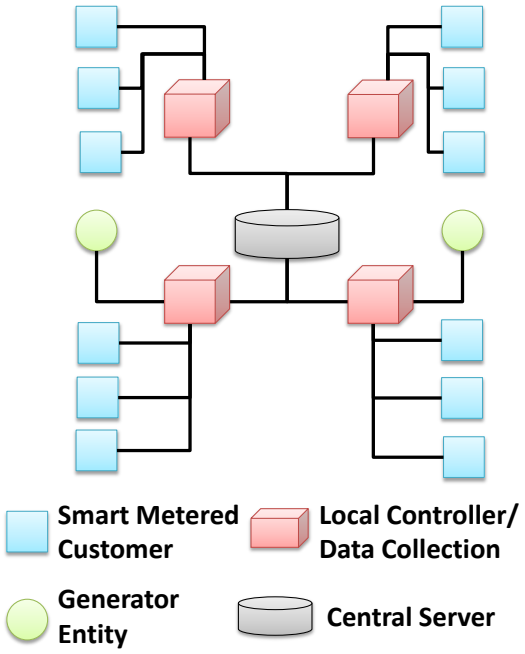


Fig. 7.2 – Typical Smart Grid Communication Architecture

The four centrally located agents were considered to be part of the central server entity of the architecture. The control was also within the aggregation tier of the hierarchy to represent the common concept of local controllers when operating as a static architecture the local controllers could only influence customer agents on the same feeder as the controller. Contrastingly the initialisation stage created a more distributed communication structure whereby customers were able to select which controller they preferred based on connectivity. Therefore customers who are connected on the same feeder were not necessarily under the observation of the same controller. This process was applied as the initialisation stage was considered to be part of the self-organising sequence of events and therefore would not be applied to the static architecture.

7.4 CONTROL SCENARIO

Each of the customer and generation agents was supplied with a profile dictating its demand or generation at any time during the course of the simulation. From the perspective of the customer population the objective of the profiles was to create a voltage deviation event which presented with long enough duration to create a control problem which required continual monitoring by the controllers. On the basis that the simulation period considered 20 minutes of system operation, the load profiles focussed on exacerbating a period of peak demand. This could be a reflection of a series of electric vehicles being connected for charging or activation of heat pumps within the customer premises.

Each customer was responsible for the monitoring of its own voltage level, which reflected the voltage of the network bus which the customer was connected to. A persistent voltage deviation was reported a customer's associated controller which triggered the control process. The actual control process was delivered through demand side response in the form of load shedding actions. A percentage of the population would reduce their demand by 700W when instructed by a controller. If a customer was placed in a state of load shedding it would send a series of messages to the controller to ask whether or not it could relax the controls and return to its nominal profile.

The demand restriction process created a rise in the voltage profile for each of the customers, once the voltage at the tail of the feeder was raised above a threshold value of 0.96 per unit, the controller would respond to the check messages sent by those customers under control to inform them that they could release any restrictions put in place.

7.5 ATTACK CONDITIONS

The primary attack method used throughout the testing process was based on a denial of service (DoS) attack created through a large amount of unhelpful messages being transmitted to target agents. The aim of a denial of service attack was to flood the target messages which served no purpose such that conventional operation was interrupted or slowed as a result. In the context of a monitoring and control system present within a smart grid environment, a denial of service attack has the potential to hold up or even prevent control signals from being delivered. The attack also has the potential to create a significant backlog of information which prevents information in the form of control requests and measurements from reaching their desired destination.

During the evaluation process the DoS attack was hosted at the customer layer as the volume of smart-meters provided a suitable launch platform for the attack event. Furthermore the accessibility of the smart-meter hardware presented a system vulnerability which could be accessed by a wider range of potential attackers. When an attack event was triggered at least one of the customer agents transmitted a large volume of information at the aggregate agent it was associated with. The target aggregate was tasked with performing control actions in addition to data collection tasks; therefore the goal of the attacker was to interrupt the control interaction between customer and aggregate and prevent control actions taking place. To ensure the attack event remained within the bounds of a real-world example the effective data output from a smart-meter performing an attack was curtailed such that it didn't exceed the transmission capabilities of a 2.4 GHz ZigBee transmitter. This transmission technology is planned for the widespread smart-meter rollout within the UK, and even taking into account that future technologies may have much higher transmission rates, legacy systems will remain connected and limited by their hardware. The attacking agent had to share this available transmission capacity with the other core functions of the customer agent which include communicating demand information and interacting with the gateway agent.

The following attack formats were applied to the static and self-organising architectures, in a series of increasing attack populations:

7.5.1 Burst Attack

The burst attack approach involved a short period of time within which the attack traffic was transmitted, each of the events lasted 250 seconds and was timed to coincide with the

initial request for control. The aim of the attack was to interrupt the control requests and therefore reduce the chance that the controller would be aware of an ongoing voltage deviation event. Each of the attackers involved in the burst event were triggered simultaneously by the error generation agent to concentrate the actions of all attackers. An attack of this nature would represent a probing event by an adversary, testing the capabilities of the target network.

7.5.2 Sequential Attack

A sequential attack consisted of a series burst attacks across the lifetime of the simulation each one individual event matching the format of the original burst attack. The sequential attack strategy had the objective of performing multiple probing stages each to test the architecture after transitional decisions by the architect had been implemented. Over the course of a simulation run, three instances of a burst attack were performed with 200 seconds between each of the events.

7.5.3 Continuous Attack

The continuous attack, once it was triggered, lasted for the remainder of the simulation and therefore the attack traffic was delivered for a substantially longer period of time in comparison to the burst attack incidents. As with the previous two attack strategies the start of the attack was designed to coincide with the start of the control process such that once the congestion created by the attack started to build, the architecture was in the process of disseminating control signals. However the control mechanism as documented in a later subsection requires continual interaction between controller and customer. The continuous attack aimed to be able to sever that interaction and therefore prematurely cause customer agents to release demand restrictions placed upon them by the controller.

7.5.4 Static Attacks

Each of the attack strategies was implemented in two formats, the first of which being a static attack – this method ignored the actions of the architect agent and did not respond to changes within the architecture. The attacker, or attackers, received a message triggering the start of the attack event, and sent the volume of attack traffic to the aggregate it was assigned to during the initialisation phase. In the case of the static architecture, the attacker was assigned to the aggregate associated with the feeder it was connected to. Regardless of the actions taken by the architect agent, throughout the duration of the attack the messages were always transmitted to the same target. If the architect agent decided to remove the

target from the population by performing a substitution action, the attacker would then be transmitting the attack traffic to a dormant agent, thus isolating the attack stream. This was indicative of a low sophistication attack strategy, and one which assumed the attacker considered a static network of agents.

7.5.5 Adaptive Attacks

The second of the attack implementations was the adaptive attack format; this represented an escalation in sophistication and one which responded to the decisions made by the self-organising architecture. In the adaptive format the initiation of the attack remained the same, as an attacker would transmit the attack stream to the aggregate it was associated with as a result of the initialisation phase. However if the architect decided to perform a transition event, and the attacker became associated with a different aggregate – either due to a relocation or due to a substitution event – the attacker would then redirect the attack traffic towards the new aggregate. The adaptive attack mechanism monitored the agent to which legitimate traffic was being transmitted following a transition event and adapted the target of the attack traffic accordingly.

7.6 TEST ENVIRONMENT DESCRIPTION

To simulate the combination of the agent population and the electrical network a test environment was developed, this environment was introduced to remove the number of hard coded variables added to the first series of tests. This was due to a greater degree of flexibility being required to transition between architectures. The sequential voltage calculations were replaced with an external load flow engine, which was supplied with a model matching the test network configuration presented in a previous sub-section.

Due to its accessibility and applicability to the interfacing issue Matpower was selected as the eternal load flow engine of choice. Using a load-flow engine lead to fewer restrictions in the communication process between agents in the MAS, because under the former system several messages had to follow the flow of electricity through the network in order to ensure that the calculations retained accuracy and continuity. Therefore it provided more opportunities for network freedom and preventing relocation of agents within the MAS compromising the flow of information required to perform voltage calculations. If a customer was to be relocated under the previous system or an aggregate agent replaced by its substitute, restructuring the flow of messages involved in the sequential voltage

calculation would create unnecessary complications and potentially compromise the accuracy of the information generated by those calculations. Eliminating this concern by separating the electrical calculations from the roles and responsibilities of the agent population prevents this issue from occurring.

7.6.1 Setup and Configuration

The process of connecting the two components is discussed below documenting the additional files, data and steps needed to be able to share information between the two platforms. The following additional files needed to be included in the process.

Required Files and Agents

To develop the interface between the two components, a series of files were required to facilitate the process of sharing information between two systems which had no direct communication route. The additional files bridged the gap between the agents in JADE and the load flow engine in Matlab.

- **Gateway Agent (GA) –** The GA was an additional agent which is introduced to the population for the purposes of handling the interaction between the Matpower load flow engine and the rest of the MAS. This agent performed several roles within the architecture, initially the GA processes the components file and passes parameters out to the agents. Parameters, such as voltage limits, ratings and initial set points, such that the information held by the agent population matches the information present within the model file. After this data had been disseminated, the GA then turned its attention to triggering the load flow engine and waited for the results to be returned. It had to check for the existence of the results file which was cleared after each calculation and check for the end of file tag, thus confirming that the load flow stage was complete. The remaining duty of the GA was to retrieve network information from the agent population, customer demand, generation information and use that to re-write the Matpower model such that the subsequent load flow was representative of the most recent data from the agent side of the simulation.

- **Components File:** The components file was a csv document containing all the network components, connections and busses – along with each a set of parameters for each of those elements. This file allowed the GA to map the set of agent names

used by Jade as a method of distinguishing between members of the population to the component ID numbers used as part of the Matpower model file format. Therefore when the GA was presented with a set of bus voltages, these can then be passed onto the agent representing the component connected to that bus. In addition to assisting identifying bus ID numbers the component file was also of use to the architect agent providing overall network information which is then later used in the process of validating the initialization stage and performing architectural transitions.

- **Matpower Model File –** The Matpower file contained the data and parameters for network described in section 7.2, two versions of the Matpower file are used for the purposes of the simulation. One version is the original starting configuration and the template which was used by the GA as a base to re-write the active version of the model. This active file was the file which was accessed by the Matpower load flow engine to perform the load flow calculations.

- **Batch File –** To trigger a load from the GA, the agent needed to call Matlab from the command line, and therefore performed this via the use of a batch script containing several commands which navigate to the Matlab root directory activating a minimal version of the application loading a Matlab script to manage the load flow calculation process.

- **Matlab Script File –** Once triggered, Matlab was instructed to run a specific script which calls the Matpower load flow engine and loads the model file. In addition to performing the load flow phase of the simulation the script file is also responsible to handling the results of a load flow operation. The script was also responsible for retrieving the results from the calculation and populating an output file with the data, formatted so that it was easily parsed by the GA.

- **Results File –** The final component of the interface between Jade and Matlab is the output from the load flow calculation, this file contains a list of bus ID numbers taken from the model file and their corresponding voltage information. This file is read by the gateway agent, and agents which have a corresponding network bus will be sent the most recent updates from the power simulation side of the process.

The documented set of files and agents, operated across a series of steps during the course of a simulation, the java side of the system remained in control over any load flow calculations. In addition to the set of files other preparations were required before initiating a simulation event, these addition stages are described as follows.

Building the Network Model

Prior to operating the self-organising architecture and running the simulations the network model was developed in IPSA2, this software package was utilised at this stage for the ability to visualise the structure of the network. It was also useful run a series of tests to ensure that the load and generation profiles delivered a voltage profile which contained a deviation event for the control system to tackle and that those deviations do not create insurmountable control issues. Finally the software allowed additional information to be added to the underlying diagram, information which aided the overall simulation. For example individual components could be named, so loads were given a name which corresponded with the agent name present within the agent architecture. The source code for the model separates different components and lists all their parameters; these parameters were then extracted such that they could be used to build the components file, allowing the GA to identify how many loads are present and their associated bus ID number.

Building the Components File

The second preliminary stage was the creation of the components file which acted as a directory of the components and their associated agents which was used by the gateway agent to ensure the correct load flow results are disseminated to the corresponding agents. The components file as previously discussed was a CSV file containing listing of all the relevant entities present in the network model combined with agent information regarding the initial starting state of the aggregate population. This file was constructed from IPSA model source file which documented each of the component types and their list of parameters – therefore the names applied to the components when drawing out the original model are associated with their component identifier as well as the identifier for the bus they are connected to.

Building the Matpower model

A third preparation stage was the conversion process from the initial IPSA model file into the modelling format used by Matpower – as both file formats can be accessed in a raw text format the conversion script read the IPSA source file and extracted the pertinent

information. This information was then re-written into the format dictated by Matpower – during the conversion process not all of the information contained with the IPSA file was retained. For example there was no separate format for load components as the information was recorded at the bus level rather than the component level. Furthermore all components are identified by a numerical component index as Matpower does not record any of the component names used in IPSA. It was this change in the information format which defines the need for the components file such that the GA had the ability to understand which agent each of the component identifiers belongs to.

Building the Agent Population

The final stage was to manufacture the agent population, given the large number of components represented in the network and therefore warranting agents within the self-organising architecture it was not practical to manually compose each of the agent files. Additionally because each customer, generator or aggregation agent may have differing parameters based on the network configuration it was also not necessarily feasible to clone a series of agents based on a given agent type. As a result a procedure for mass producing an agent population was also required.

This process used agent template files as documented in the appendix section, which consisted of the entire agent source code with the variables replaced by placeholder code words. An agent manufacturing program written in java would then read through the components file and determine how many copies of each agent file to make and which parameters to replace the code words with. Other parameters within the manufacturing program could also be set to dictate the operation of the simulation – such as selecting load and generation profiles from a collection of potential data sets, and controlling profile compression rates.

Producing a population of 340 customers, 4 PV generators and 8 aggregation agents using this process would customarily take around 20 minutes to complete. Additional agents such as the GA, Observer and Architect agents did not need to be created in this process as they did not require manufacturing in bulk quantities.

7.6.2 Running the Simulation

With the all of the files and agents created, the remainder the interaction between the Matpower load flow engine and the agent population took place during the runtime of the

simulation and iterates over a series of stages. The following stages document a simulation running without the intervention of an attack event or control actions – focussing on the operations required to connect an agent population created in java with an external load flow mechanism.

Step 1 – Forming the Knowledge Base

The first stage runs acts as an initialisation stage for the interface and is centred on the GA and its responsibility to perform the association between the numerical bus identification format used by Matpower and the name based format of the agent platform. To perform this task the GA processed the components file which contained the network model information alongside the relevant agent names the information corresponds to. The GA built a knowledge base as an object-oriented representation of the network model, this knowledge base is then later used to store up to date parameters received from load flow calculations. The UML diagram in Fig. 7.3 documents the structure of the knowledge base.

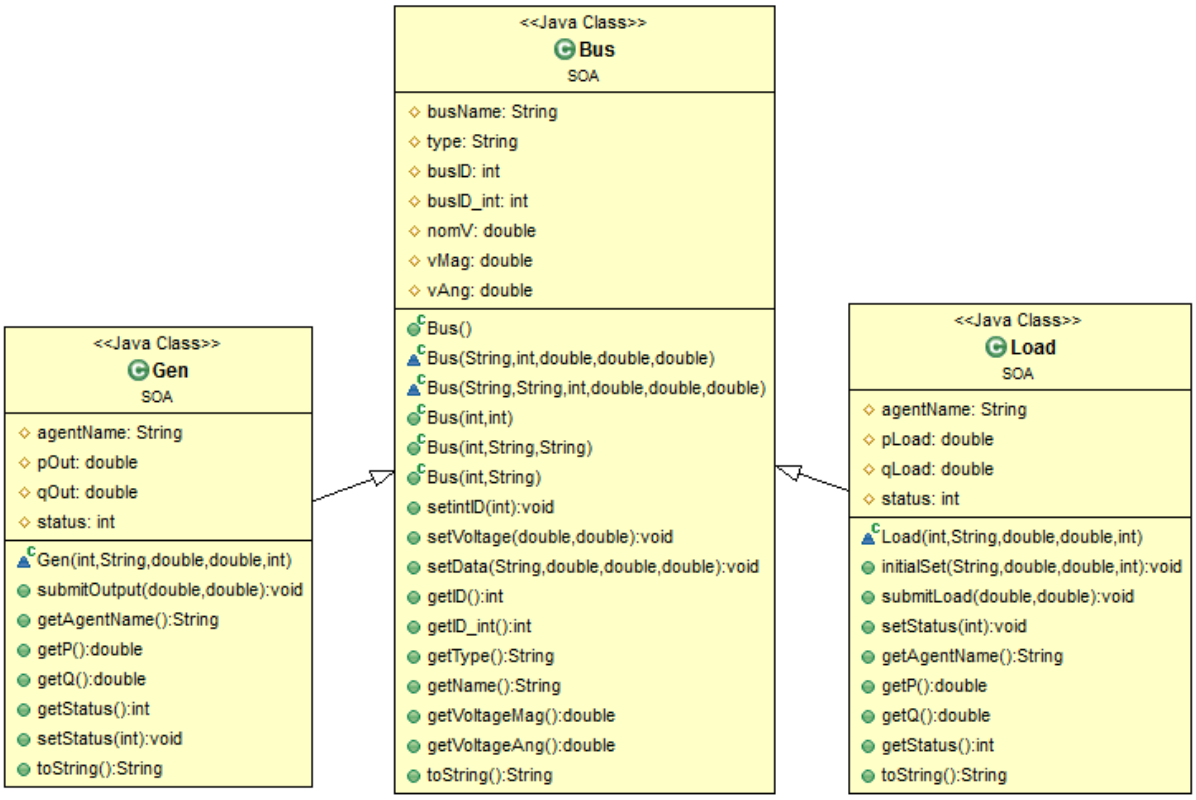


Fig. 7.3 – Gateway Agent Knowledge Base

Step 2 – Retrieve Agent Data

Once the knowledge base was complete the GA began listening for agent updates in the form of customer demand and generation agent output data. Each customer was supplied with a load profile, to ensure that the total demand varied over time, setting the conditions

for an under-voltage event which triggers a control response. This data collection process is completed through the agent network, communicated via a series of Agent Communication Language (ACL) messages. In this stage the knowledge base acted as a lookup service, as the GA received an update from a member of the agent population it updates the knowledge base so that it can then be used to write a new iteration of the Matpower model file and thus receive load flow information for the current system state.

Step 3 – Re-write the Matpower Model

The process of writing the Matpower model file was part of the core loop which controls the interaction, every three seconds the GA pulls the information from the knowledge base which is continually updated and writes a new Matpower model. To do this the GA read in the original version of the Matpower file one line at a time, and lines such as headers, section start and end tags, and busses without connected load or generation were copied directly to the new file. Other lines needed to be modified for the purposes of injecting the most recent data from the knowledge base; these lines were broken into an array of strings to make the individual fields accessible. The majority of the fields remained unchanged, but customer demand or generation output figures were replaced before the array was concatenated into a single line so it could then be transplanted into the new file. Finally an 'end of file' tag is added to the end of the new file such that the GA could determine when the file writing process has been completed.

Step 4 – Executing the Model

With the new model file completed it was then important to execute a load flow calculation and retrieve the results. This was performed through the provision of a file entitled "matcall.bat" containing a series of command line instructions to navigate to the root directory of the Matlab application and execute a Matlab script. The contents of this batch file are presented in Fig. 7.4, where several runtime arguments are passed to Matlab when it was executed to ensure that the most minimal version of the engine was triggered. These arguments prevented Matlab from opening a full version of the application and only accessed the core engine to run the script triggering a load flow calculation.

```
C:
cd/
cd Program Files
cd Matlab
cd r2012a
cd bin
matlab.exe -nodisplay -nosplash -nodesktop -minimise -r
run('H:\MATLAB\runLoadFlow.m');exit;
```

Fig. 7.4 – Batch File

The Matlab script triggered by the batch file loaded the Matpower model written in step 3 and performs the load flow calculation. This script was also responsible for the creation of the result file which detailed all the voltage magnitudes for each of the buses in the network. This information was formatted such that it could later be read by the GA once the load flow process is complete. At this stage the GA agent itself was placed on hold, scanning for the result file before it can continue to the next stage, it would however still be receiving information from the agent population and updating the knowledge base so that the next model file writing stage contains the most relevant information.

```
output_file =
'C:\Users\A7178941\EclipseProjects\SelfOrganised\results.csv';
mpc =loadcase('ModelFileTMP.m');
myopt = mpoption('verbose', 0, 'out.all', 0);
define_constants;
results = runpf(mpc,myopt);

buses = [results.bus(:,1), results.bus(:,8), results.bus(:,9)];

dlmwrite(output_file,buses,'delimiter',',','newline','pc');
results_file  = fopen(output_file, 'a');
fprintf(results_file, '%s', 'File End');
fclose all;
```

Fig. 7.5 – Matlab Script

At the end of the result file is another file end tag, it is this tag that the GA was looking for before moving onto the next stage of reading the file. These tags were put in place to avoid the GA from prematurely reading the result file and attempting to parse incomplete lines of text which in turn would create problems for the GA itself. Once this tag was discovered by the GA, it knew that the load flow stage of the loop was complete and it began reading the results file and disseminating the information.

Step 6 – Reading the results

The final stage of the main loop was the reading of the of the results file, in this instance the GA processed the file line by line, where each line contains three pieces of information

Bus ID, voltage magnitude and voltage angle. This information was transferred into the knowledge base maintained by the GA using the bus ID as the index value, once the end of the file had been reached the GA would then go through the knowledge base and retrieve the data for all busses with associated generator or load components connected. Each of these components was then sent an ACL message containing the results of the load flow for the purposes of performing voltage monitoring

7.7 IMPLIMENTING CONTROL

To perform this control function the customer agent observing the voltage deviation needs to contact the controller and indicate that a control decision is required. The controller however only has a limited number of agents it can request a control response from, based on those agents which connected to it during the initialisation stage. Therefore the controller assesses the connected population and determines which of those agents are electrically nearest to the customer reporting the deviation. This process involves communicating with the Architect agent to retrieve a list of agents on the same feeder as the control request; this list is compared with the controller's list of connections. Those agents who fall in both lists are both applicable to the deviation and accessible by the controller, initially four agents from this list of matches are contacted, and asked whether or not they will accept a control command. If the agent will not accept the command it is registered as uncontrollable and will not be contacted in the event of future control events. Otherwise the controller will ask the agent to apply a control restriction, if the controller continues to receive reports of the ongoing voltage deviation further customers will be contacted in this manner. During the period of time in which the control restrictions are to be applied – any customer who has been issued with a restriction will repeatedly ask the controller if those restrictions are still necessary – as per the following diagram Fig. 7.6.

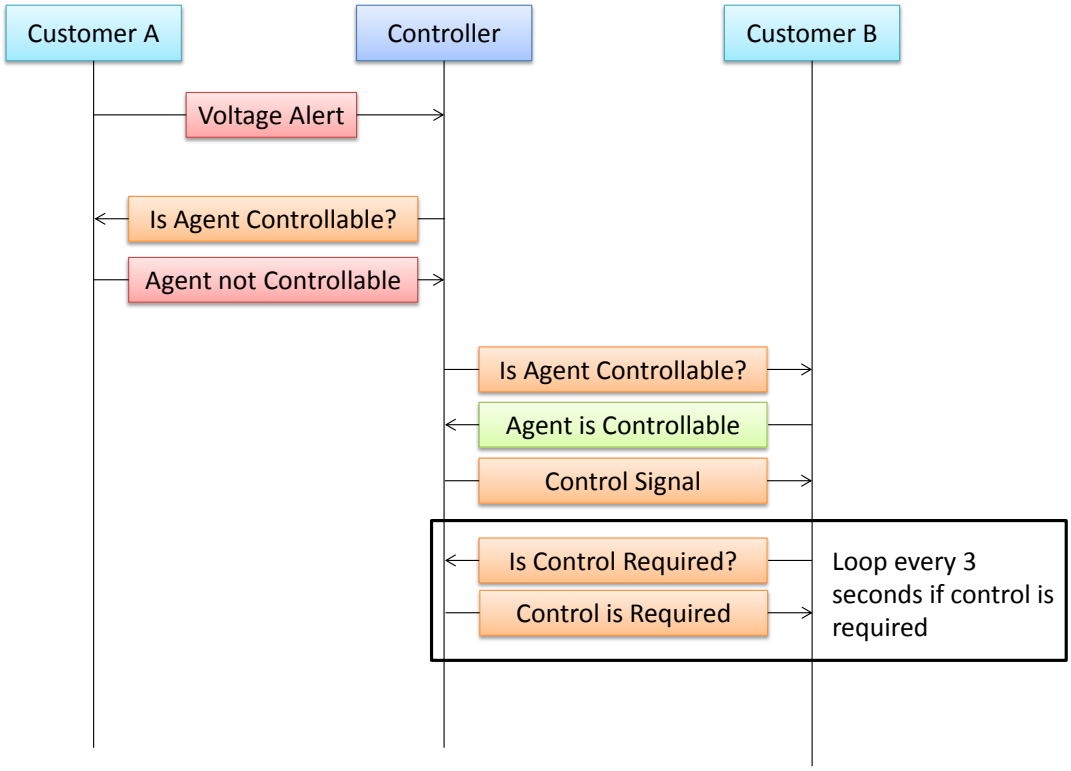


Fig. 7.6 – Communication Structure during Control

The restrictions imposed by the control process are relieved if one of two eventualities takes place, firstly if the voltage at the end of the feeder exceeds 0.96p.u. In this case the controller will respond to the 'Is control required' message informing the customer that controls can be lifted. The second eventuality is that there is a significant delay between the control query poised by the customer and the reply from the controller – the distance between the query and the response will be influenced by the communication load taking place at the controller. If the response takes more than 10 seconds – the customer will automatically lift the restrictions as it has not been instructed to do otherwise. The rationale behind customers under control continually sending query messages is to build a degree of dependency between the customer and the controller and insert system whereby the customers could return to a level of conventional consumption in the event of a controller failure.

7.8 PERFORMANCE CRITERIA

Two core points of evaluation were considered in terms of evaluating the performance of the self-organising architecture which was faced with two concurrent network issues. The first of which was the ongoing voltage deviation and the second was the attack event targeting the control tier of the architecture. The first point of comparison was focussed on

the voltage control performance, and the ability to maintain controllability during an attack event, the objective of all the attack formats was to disrupt the link between the controller and the customer and therefore reducing controllability. For example the following figure presented in Fig. 7.7, illustrates difference between a scenario with 100% controllability and zero controllability. The attacks presented within the research are aimed at creating a situation where the net controllability of the network is as close to the red line as possible.

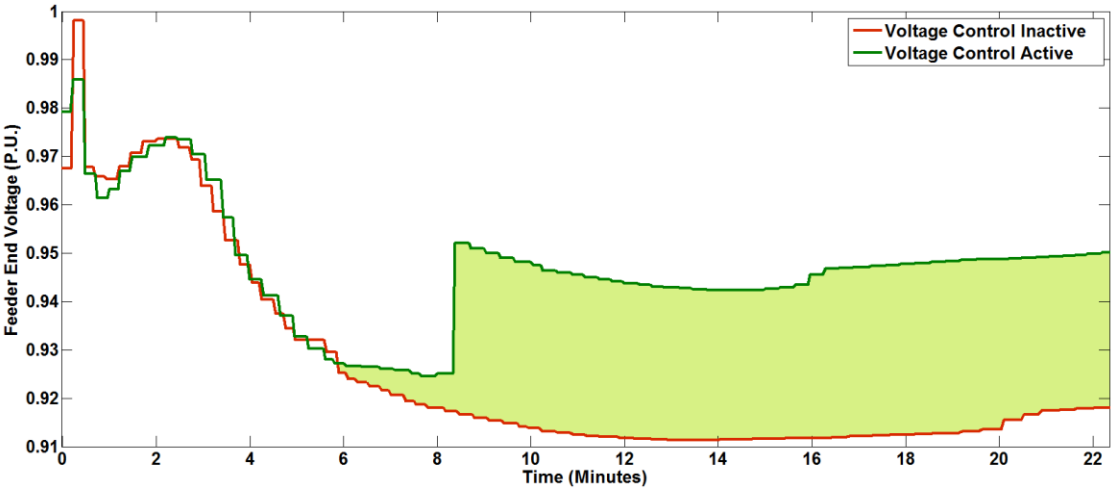


Fig. 7.7 - Controlled and Uncontrolled Voltage Profiles

Therefore during each of the simulation runs, the voltage profile from each of the agents was recorded, with the agent at the end of the feeder being representative of the lowest voltages. In contrast the self-organising architecture aimed to minimise any control loss and therefore voltage profiles taken from tests utilising the self-organising were intended to be as close to the upper threshold illustrated by the green line as possible.

The second performance criterion was concerned with the communication layer represented by the computational burden figures, these burden profiles represented the impact of the attack on the number of performance variables monitored by the architecture. In addition to the overall burden the individual components of the computation burden were also recorded, these components were as follows and as previously defined in chapter 7:

- Congestion
- Reactivity
- Data Flow (Incoming and Outgoing)
- Unresponsiveness
- Execution times
- Under-utilisation

Individual burden figures were calculated through totalling the error severities for a given error report type and then dividing that figure by the number of agents which could be responsible for creating the error. The reason why the additional performance figures are calculated was to illustrate the wider performance gains delivered by the self-organising architecture. In cases where the voltage profiles indicated a small improvement in terms of the control performance, when the additional communication data was examined wider implications of the architecture could be investigated. The ability to reduce the burden from the perspective of reactivity or congestion would indicate that the self-organising architecture can demonstrate properties which would be relevant when dealing with control problems with shorter time resolutions – frequency response or protection systems for example. Likewise being able to reduce the amount of incoming data present at key bottleneck areas in the architecture would be beneficial in systems which are energy sensitive. For example if the aggregate agent was not connected to a permanent power supply, reducing the communicative load makes the agent more energy efficient and therefore extends its lifespan on the power available.

To demonstrate the monitoring capabilities within respect to computational burden, the following figure presented in Fig. 7.8 illustrates an example performance test in the absence of an attack event.

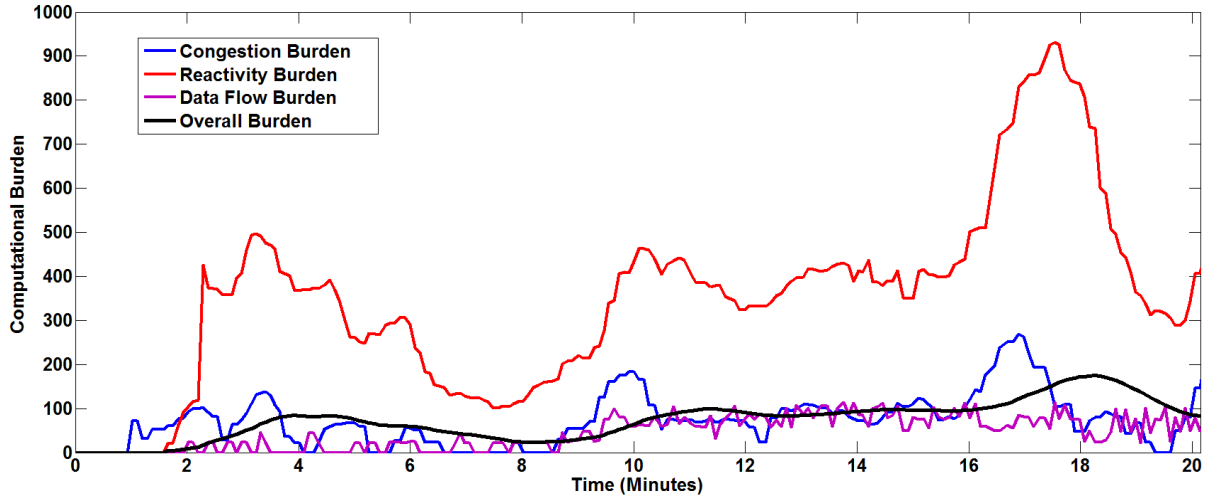


Fig. 7.8 – Computational Burden Composition Example

Four profiles are presented in the figure to demonstrate the range of profiling performed during a simulation; the first three consider individual properties of the architecture as the simulation progressed. The fourth profile represents an overall burden figure, and the value which was used as the input into the fuzzy decision making engine. A series of profiles was

recorded from each of the attack format tests with respect to both the static and self-organising architectures such that they could be compared.

7.9 SUMMARY

This chapter documented the configuration and processes required in running and operating the simulation. Initially the chapter discussed the design and properties of the electrical network considering the location of the associated agents for the components involved. This was followed by documenting the nature of the agent architecture and the differences between the static and self-organising architectures from the perspective of performing the evaluation process. Secondly the chapter considered the requirements for setting up a simulation event, taking into account the different source files and supporting files needed to perform the simulation. Additional taking into account the processes involved in linking the agent architecture based on the JADE agent platform with an external load flow engine for the purposes of performing voltage calculations.

The chapter also documented the attack approaches which are to be applied to the architectures to test the effectiveness of the decision making engine and the transitions triggered in response to the attack event. Finally the chapter considers the points of analysis with respect to the architecture performance in terms of both the voltage control process and the communication layer performance.

Chapter 8: Results

8.1 INTRODUCTION

The static and self-organising architecture were tested against a series of denial of service attacks launched by the customer layer against the set of controllers. The majority of the attacks, once triggered lasted for the length of the simulation and were declared to be continuous. This format was selected for most of the attacks because they represented the most severe instance of each of the attack scales involving increasing numbers of customers. Further attacks consider intermittent attacks in the form of burst and sequential denial of service events. During the course of a simulation each customer was supplied with a demand profile which created an under-voltage event, this event would persist without control intervention. Therefore in addition to the presence of a cyber-attack the self-organising architecture would also have the challenge of performing voltage control through the use of demand side response. As previously discussed the control process was performed through the transmission of control request and action signals between the controller and the customer agent. The role of the attack event was to prevent these signals from being interpreted and therefore responded to. If a customer fails to receive a response from the controller informing it that the demand side restrictions are still required, the customer will cease performing control.

This chapter presents the results from a range of attacks performed against a static and self-organising architecture. In addition to documenting the attack events, a set of results are also presented which document the performance of the architectures in the absence of control and in the absence of an attack event. This chapter is not an exhaustive documentation of all test formats considered and focusses on the continuous attacks which are presented in greater depth. All results examinations are discussed at the end of the chapter and therefore used to evaluate the overall effectiveness of the self-organising architecture under examination.

8.2 BASELINE PERFORMANCE

As a starting point the simulations were completed without the influence of an attack situation and without the control procedure disabled. The purpose of these simulations was to demonstrate that the architectures could perform the core control objective and to illustrate a level of baseline performance against which later simulations could be compared against.

8.2.1 Without Control

The first of the base line tests considered the architecture operating without any controllability, therefore illustrating the consequences of total control deterioration. This state was the ultimate objective of the attack event, as the stream of messages delivered by attackers within the customer layer intended to sever the connection between controller and customer. In each instance the voltage profile was extracted from the final customer on each feeder as it was this location whereby the impacts of an attack would be most significant. The following graph presented in Fig. 8. illustrates the voltage profiles of all four feeders when the controllers are deactivated and only function as data collection agents.

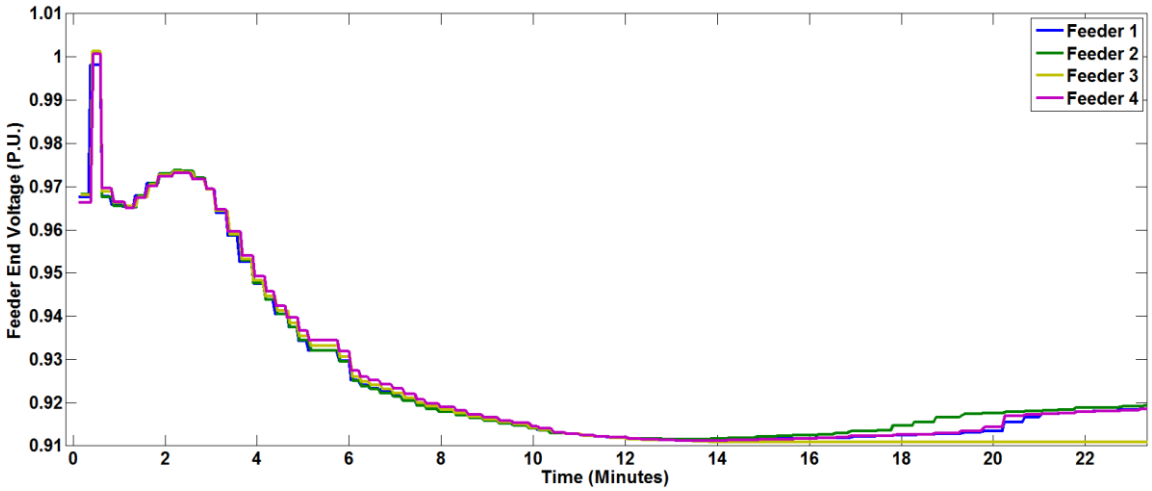


Fig. 8.1 – Voltage Profiles without Control

With control disabled the voltage on each of the feeders falls below the lower threshold and fails to recover, indicating the severity of the voltage deviation triggered by the demand profiles embedded in the customer layer. The individual profiles exhibited slight differences on account of the location of the customer from which the data was extracted within the overall network. As second figure presented in Fig. 8.2 illustrates the message congestion taking place during the course of the simulation.

The figure illustrates that the congestion was similar across all four of the aggregate agents which functioned as controllers. The data referred to the sampled queue size measurements rather than the average data and therefore indicated that congestion in an architecture with no additional functionality other than data collection peaked at 120 messages with most events containing fewer than 80 messages.

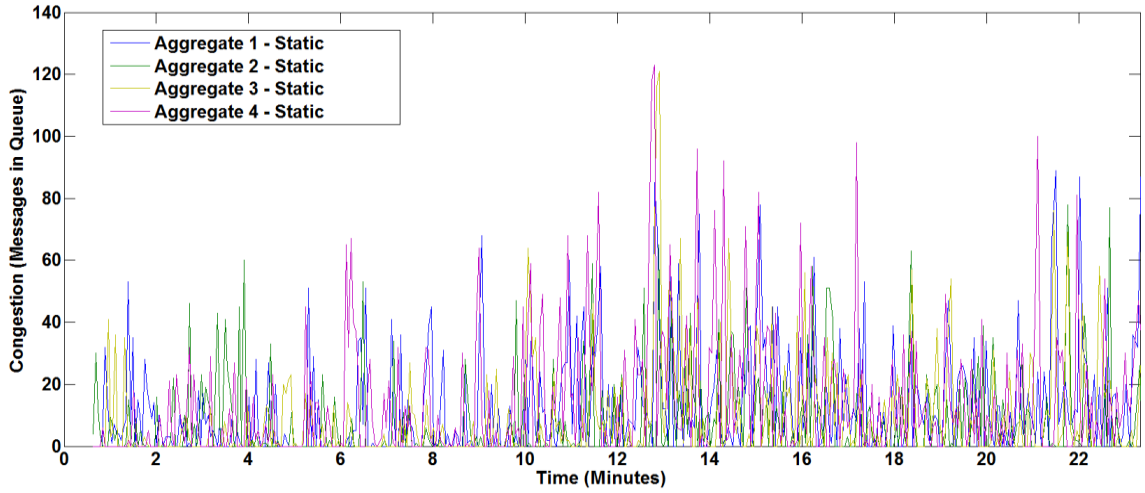


Fig. 8.2 – Aggregation Congestion without Control

Furthermore as the figure represents the congestion as a series of individual events it illustrates that those congestion events were not persistent and the aggregates were able to clear their backlog of messages quickly.

8.2.2 With Control

A second baseline comparison was to re-introduce the control functionality but still remove the cyber threat element of the investigation; the purpose of this examination was to illustrate the control performance of the architecture in the absence of an external threat. Delivering an outcome similar to the absence of any attack event was the goal of the self-organising architecture which intended to mitigate the threat of an attack and maintain controllability. In this case both the static and self-organising architectures were simulated as an assessment of the functionality without an attack. The following figure presented in Fig. 8.3 illustrates the voltage profiles under the differing architectures.

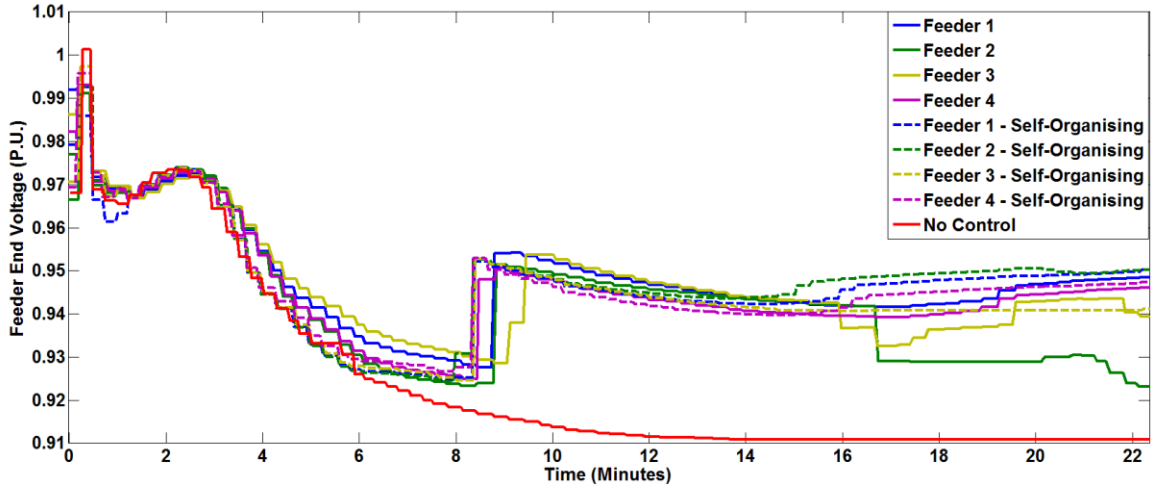


Fig. 8.3 – Voltage Profiles - No Attack

The figure illustrates that in across each of the feeders, and for both of the architectures, the control function was triggered correctly and the deviation was immediately corrected, raising the voltage above the threshold value. The self-organising architecture experienced no instances of control loss throughout the simulation whereas one feeder in the case of the static architecture did experience a degree of control deterioration. To examine the causes of this control deterioration, a second figure presented in Fig. 8.4, illustrates the computational burden indicator data recorded over the course of the simulation. The figure presents the components which contributed towards the overall assessment of computational burden in addition to the overall calculated indicator.

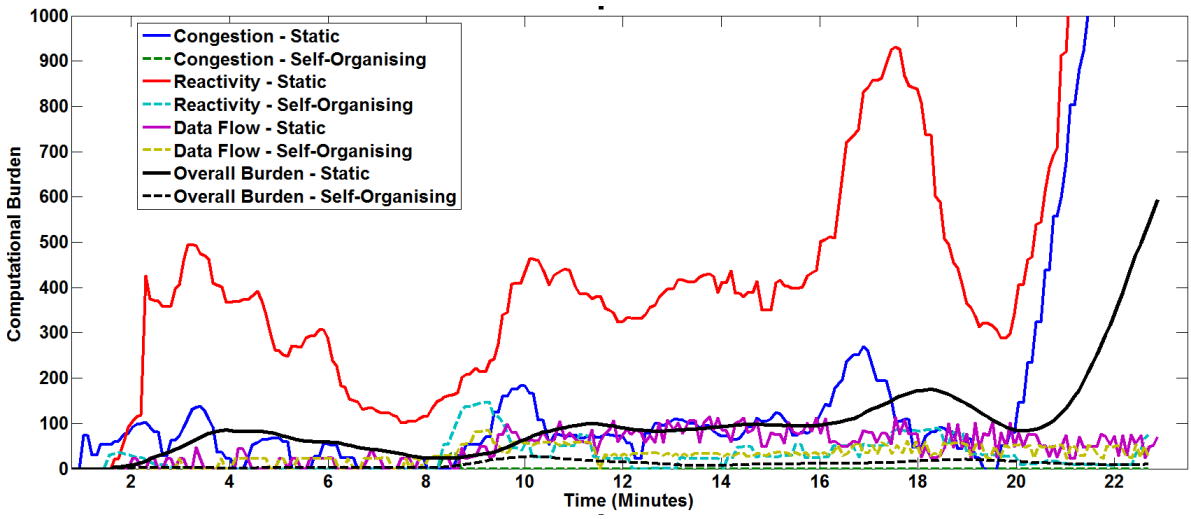


Fig. 8.4 – Computational Burden Composition and Comparison

The figure indicates that the static architecture several errors pertaining to system reactivity, however because these errors did not translate into a significantly raised burden indicator, the reactivity issue was not widespread throughout the architecture. This reactivity issue increased later in the simulation and corresponded with the control loss experienced by customers on feeder 2. As the self-organising architecture was able to avoid the occurrence of these incidents it indicated that the initialisation stage of the process delivers a stronger communication foundation to operate the architecture from as the overall burden indicator was considerably lower before any transition event would have taken place. Throughout the simulation the self-organising architecture did not experience the increases in reactivity and consequently did not observe any instances of control deterioration. The following figure presented in Fig. 8.5 examines the response times between the controller and the customer layer for the final customer on each feeder.

The figure confirms that the reactivity issues initially diagnosed within the static architecture were affiliated with the two feeders which experienced control deterioration, and therefore the communication issues were the source of the control loss. As two controllers exhibited similar communication traits is could be assumed that these controllers were experiencing difficulties in processing the volume of incoming data and therefore resulted in slow response times.

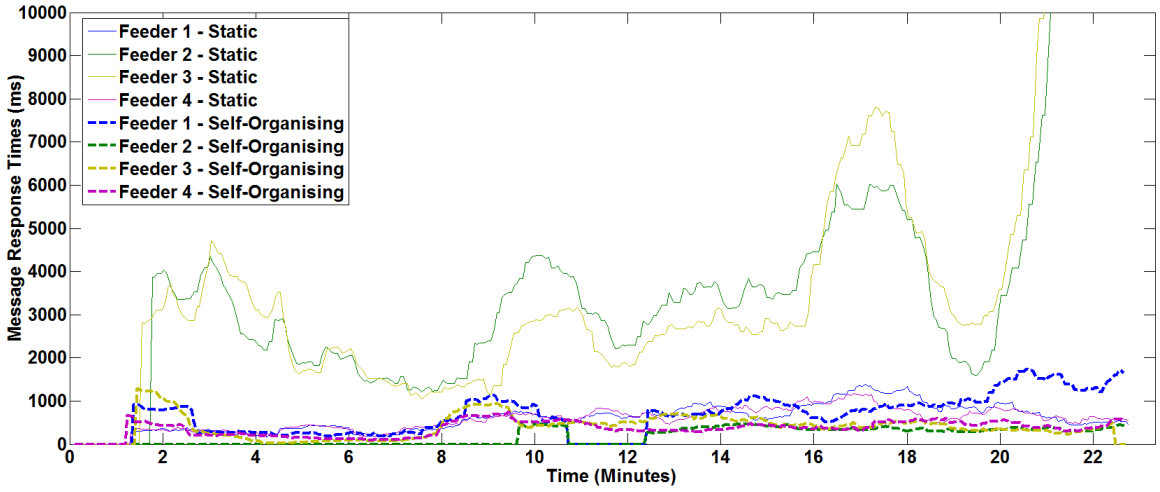


Fig. 8.5 – Response Times between Feeder-End Customer and Controller

The figure also illustrates that the response times across the aggregates in the self-organising architectures were far more uniform and all aggregates observed similar response times. In a practical implementation controllers in differing parts of the network would be exposed to variations in performance as a result of variations in component vendors, component age or external factors such as interference or weather conditions. To diagnose the source of the variation in controller agent performance, the following figure in Fig. 8.6 illustrate the incoming data flow received by each of the aggregate controllers in the case of the static and self-organising architectures.

The figure illustrates that the two feeders which observed the increase in controller response time and therefore experienced control deterioration were under the control of an aggregate which was experiencing a higher volume of incoming data. This contributed to the challenges faced by those aggregates and therefore triggered the control loss. However the figure also indicates that the self-organising architecture could lead to a more uneven distribution of communication load, as aggregate 1 observed data flow rates peaking at 0.9kB/s greater than the other aggregates.

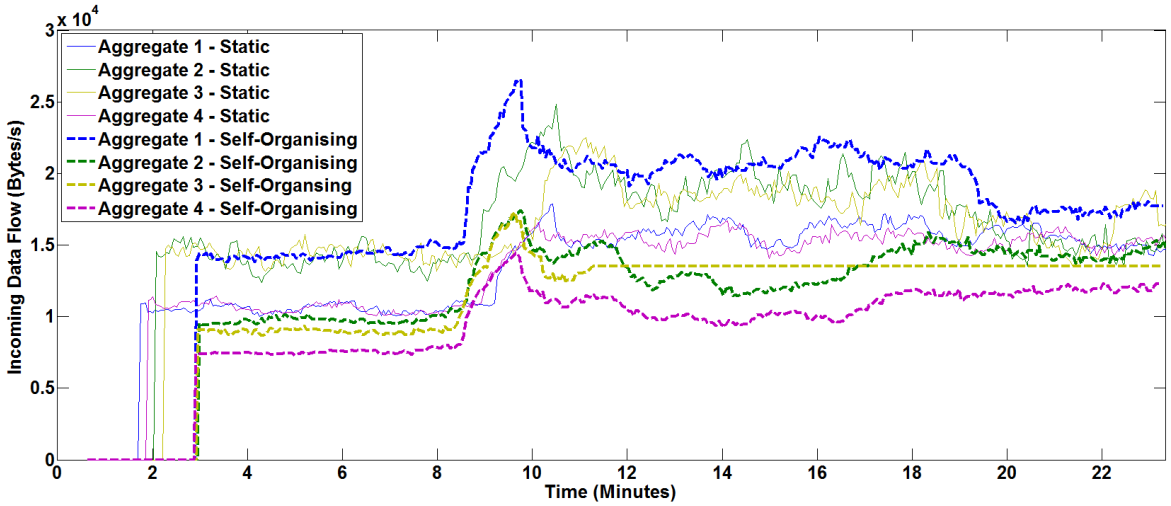


Fig. 8.6 – Incoming Data Flow at the Aggregate Layer

Furthermore the data illustrated that the self-organising architecture was able to operate with aggregates under increased load without compromising the control objective, or raising the burden indicator. A final figure presented in Fig. 8.7 illustrates the message congestion data extracted from the aggregation tier.

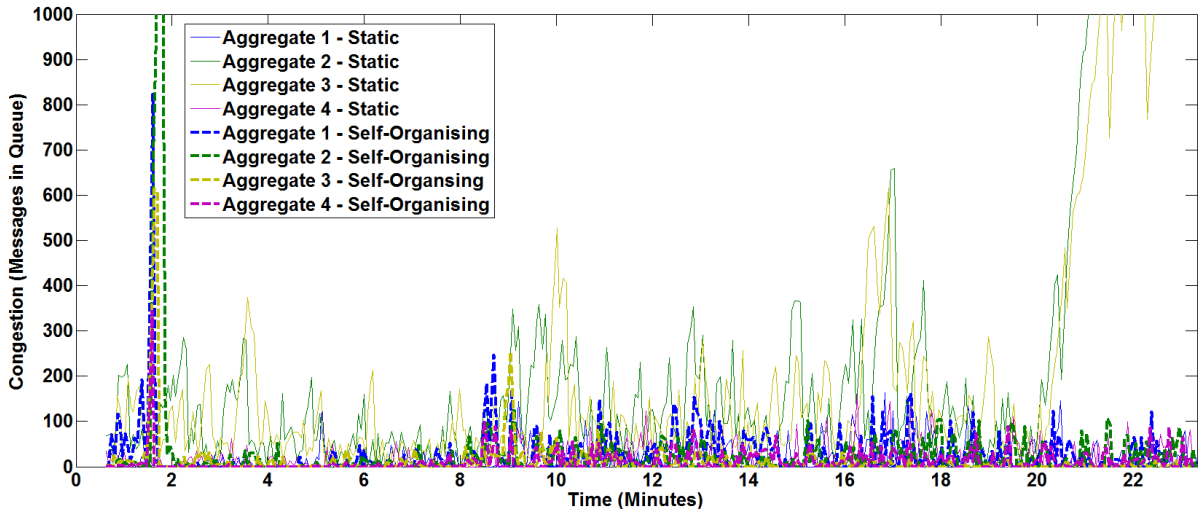


Fig. 8.7 – Congestion at the Aggregate Layer

The second figure confirms that the two of the aggregates under the static architecture collected a larger amount of messages in the message queue – and processing this queue caused the control deterioration. However an aggregate which was exposed to a larger volume of incoming data did not experience the congestion issues and therefore did not experience any control losses when the self-organising architecture was in use. This indicated that the provision of a self-organising architecture provided a more stable architecture and one which was less susceptible to variances in data production from the customer layer. Because any customers which are transmitting updates more frequently

than others are distributed across different aggregates during initialisation, their impact on the congestion and reactivity properties of the aggregation layer are reduced.

8.3 PERFORMANCE UNDER ATTACK

The second section of results discusses the performance of the self-organising architecture in the presence of the denial of service attack, in comparison with the same attack being perpetrated against the static architecture. Both static and adaptive versions of the denial of service attack were delivered using increasing numbers of attackers within the customer population.

8.3.1 Continuous attack with 6% customer involvement

The first example considers an attack whereby 20 customers are involved with performing denial of service attack at the control layer of the architecture, this accounts for 6% of the overall customer population. The attack is distributed symmetrically such that each aggregate was under attack from 5 customer agents. The voltage profiles for the static version of the attack event are presented in the following figure in Fig. 8.8

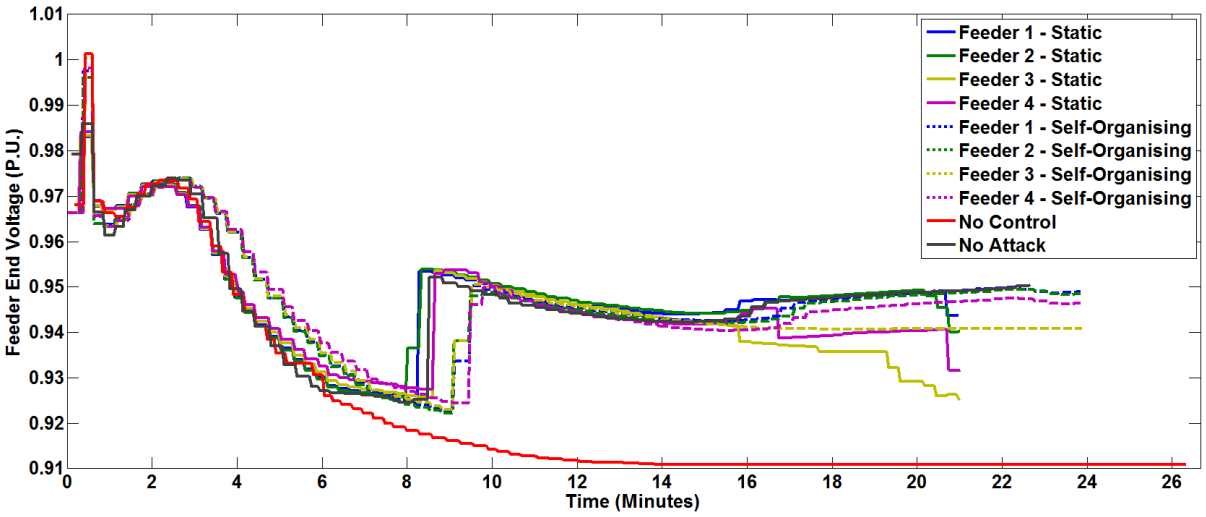


Fig. 8.8 – Voltage Profiles under Static Attack

The figure illustrates that as observed in the absence of an attack event, the static architecture is vulnerable to control deterioration even in the presence of an attack format involving a small number of customers. Two of the three feeders experienced control losses across a series of customers when operating under the static architecture. However as illustrated in the absence of an attack the self-organising architecture exhibited no such deterioration. The following figure presented in Fig. 8.9 illustrates the voltage comparisons

when responding to an adaptive variant of the denial of service attack involving the same attack population.

Fig. 8.9 uses the same voltage profiles representing the static architecture as the previous figure because in a static architecture the adaptive attack behaves identically to the static attack, and only displays adaptive properties in the presence of self-organisation.

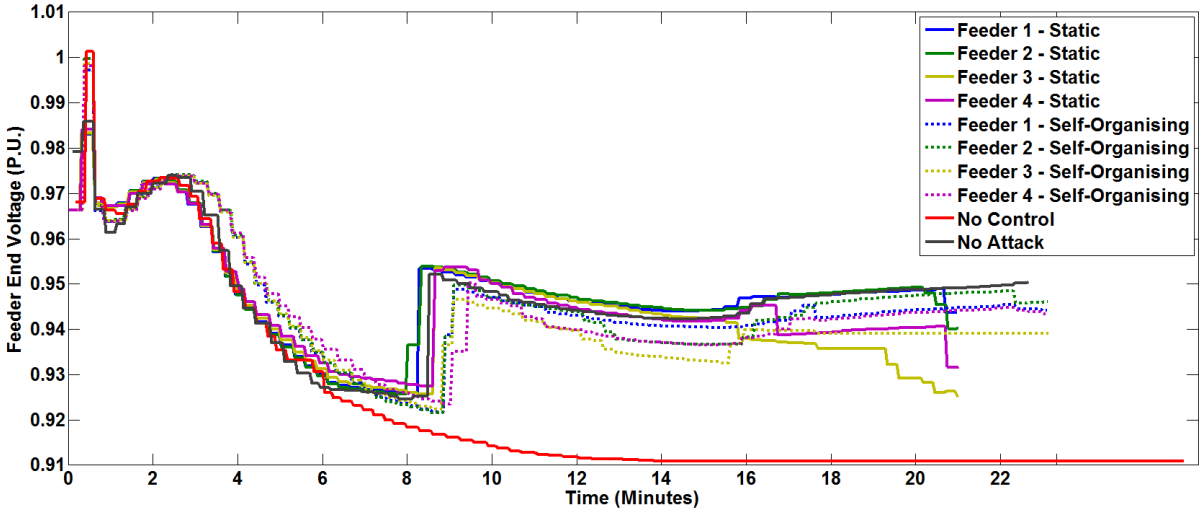


Fig. 8.9 – Voltage Profiles under an Adaptive Attack

The results of the adaptive attack are very similar to those under static attack, and therefore indicating that with 6% of customer involvement the number of attackers isn't significant for the adaptive attack mechanism to cause a greater challenge for the self-organising architecture.

In addition to the voltage profiles the attack has an impact on the communication layer of the network; the following figure in Fig. 8.10 presents the decomposition of the components of the computational burden.

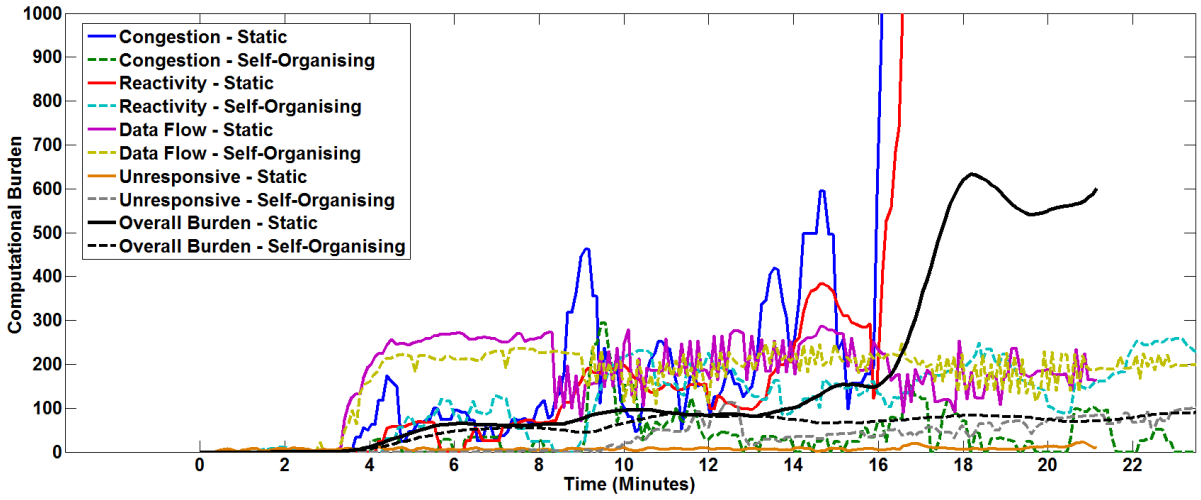


Fig. 8.10 – Computational Burden Distribution and Comparison – Static Attack

The figure illustrates that the computational burden indicator is primarily driven by the reactivity and congestion properties in terms of the static architecture. As the attack continues the severity of the congestion issues increases, and at 16 minutes they rise rapidly, this event again corresponds to the deterioration in control observed by two feeders. In terms of the self-organising architecture the primary influence on the computational burden indicator was data flow, indicating that the volume of data created by the attack population exceeded the recommended limits at the aggregation layer. This burden component remained at a similar magnitude throughout the simulation and therefore in the absence of other significant components the overall burden indicator for the self-organising architecture remained stable. Demonstrating that while the denial of service attack manifests through the production and delivery of attack traffic, which is represented by an increase in data-flow errors, it is the congestion and reactivity issues created by that data which is responsible for control deterioration. Because the self-organising architecture creates a more stable platform during initialisation, and rebalances connections when necessary, it is able to withstand the influx of attack traffic without forming congestion and reactivity problems. In both variants of the attack format the architect agent elected to activate a single dormant agent, because all of the aggregates were under-attack therefore the recommended substitution action would not be effective. The following figure in Fig. 8.11 illustrates the impact the process of activating a dormant agent had on data flow at the aggregation layer.

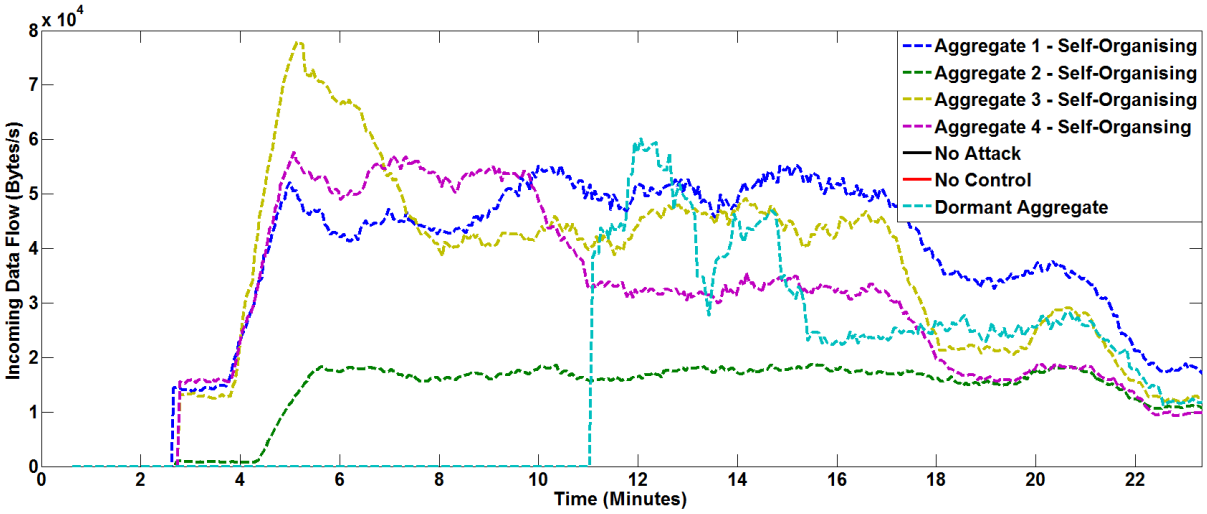


Fig. 8.11 – Data Flow at the Aggregate Layer - Adaptive Attack

The figure demonstrates that at the point where the dormant agent was activated the traffic received by aggregate four dropped by 2.2kB/s, and the dormant aggregate was exposed to the same degree of traffic aggregate 4 was prior to the instruction to activate. This was due to the nature of the adaptive attack, when a customer performing the attack was reconnected to a differing aggregate – it redirected the attack traffic to the new agent. Therefore in the case of the above figure the attackers were moved to the new aggregate, this process prevented the build-up of congestion observed by the static architecture and therefore prevented control deterioration. Another point of interest is the data flow associated with aggregate 2, which did not experience the same increase in data flow during the attack period. This was due to a rebalancing action which relocated several customers – and therefore the attack traffic was directed to other aggregate agents.

8.3.2 Continuous attack with 12% Customer Involvement

An escalation to the previous example increases the number of customers delivering the attack to 10 customers per aggregate and involving 12% of the customer population. The voltage profiles recorded during a static variant of the denial of service attack is presented in the following figure in Fig. 8.12.

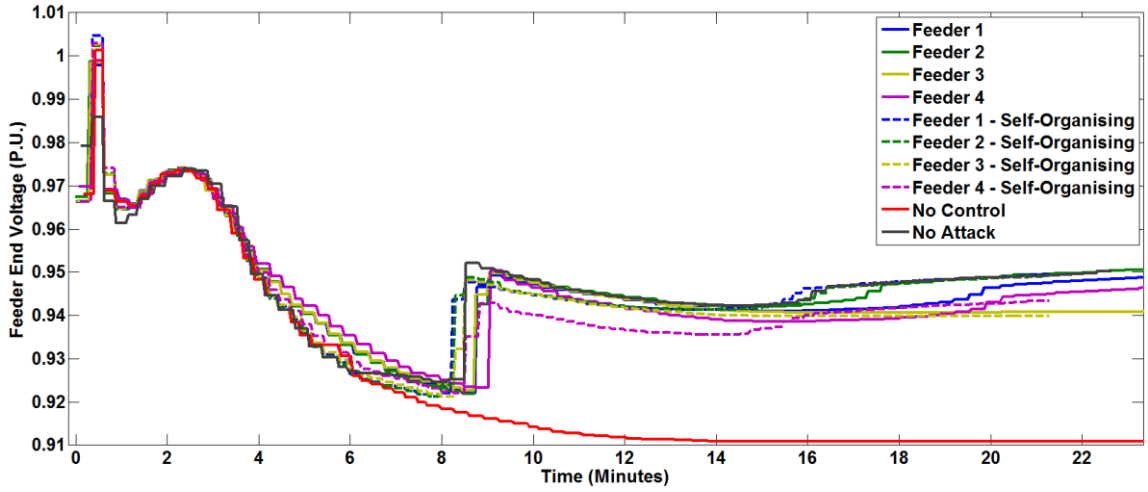


Fig. 8.12 – Voltage Comparison - Static Attack

In this example the figure illustrates that no control deterioration was experienced by the customers during the attack event, neither the static nor the self-organising architecture observed a reduction in control performance. This was also true of the adaptive attack as illustrated in Fig. 8.13. In both instances the controllability of the customer layer was not influenced by the attack, and therefore the congestion and reactivity issues which were present in test with 6% customer involvement could be related more to a configuration setting rather than the attack itself.

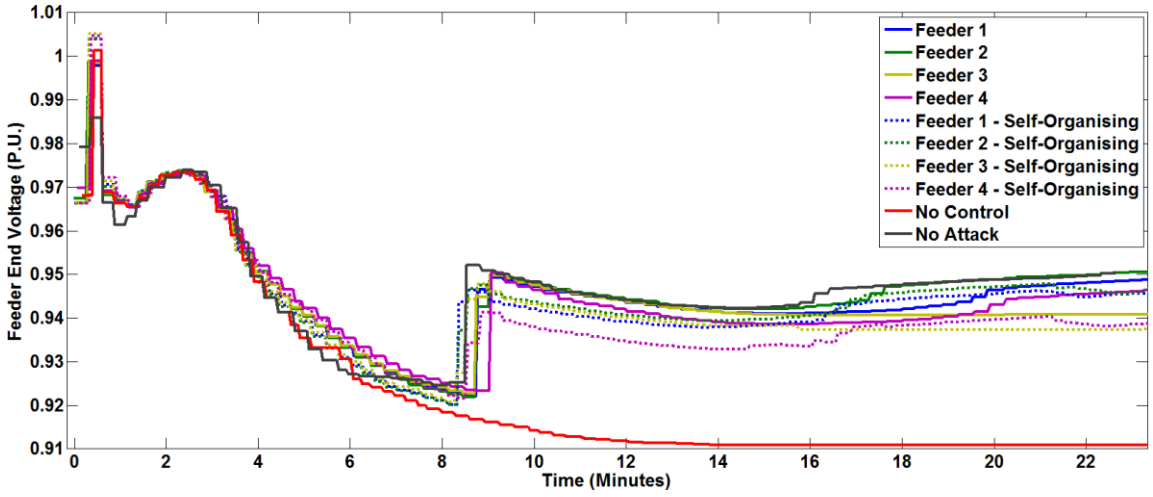


Fig. 8.13 – Voltage Profile Comparison - Adaptive Attack

This also accounts for the control deterioration experienced by the static architecture in the absence of an attack; in those cases the initialisation stage prevented an unstable starting configuration as the customer agents were able to select their own connections. Without the initialisation stage, customers are automatically assigned to the nearest aggregate which increases the chances of clustering of agents with higher outgoing data-flow rates. The self-organisation increases the agent diversity and breaks down the clustering effect.

8.3.3 Continuous attack with 18% Customer Involvement

A further escalation increased the number of customers involved in the attack to 60, with 15 customers per aggregate performing the denial of service attack. The following figure in Fig. 8.14 presents the voltage profiles recorded during the static variant of the attack.

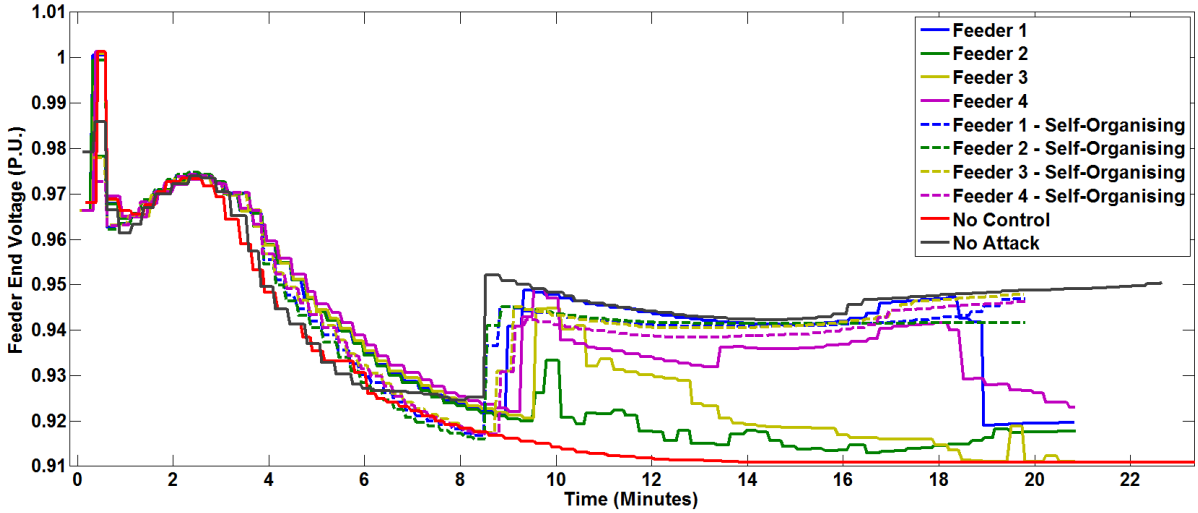


Fig. 8.14 – Voltage Profile Comparison - Static Attack

In contrast to the previous examples, the presence of the denial of service attack had a disruptive impact on the controllability under the static architecture. Two of the feeders experienced significant control deterioration, while the remaining two experienced the same deterioration later in the simulation. In contrast to the examples with fewer attackers the widespread nature of the control loss was more indicative of being triggered by the attack. The fluctuations in the voltage profiles under the static architecture – specifically in the case of feeder 2 – indicated that the customer population was attempting achieve control during the attack but the denial of service prevented a stable control connection between customer and controller. As a result consistent control over the course of the simulation could not be achieved. Additionally the self-organising architecture once again was not affected by the attack and displayed no signs of control deterioration. This example illustrated the value of the self-organising architecture from the perspective of achieving the control objective, this value was also present when processing a more sophisticated variant of the attack as illustrated in Fig. 8.15.

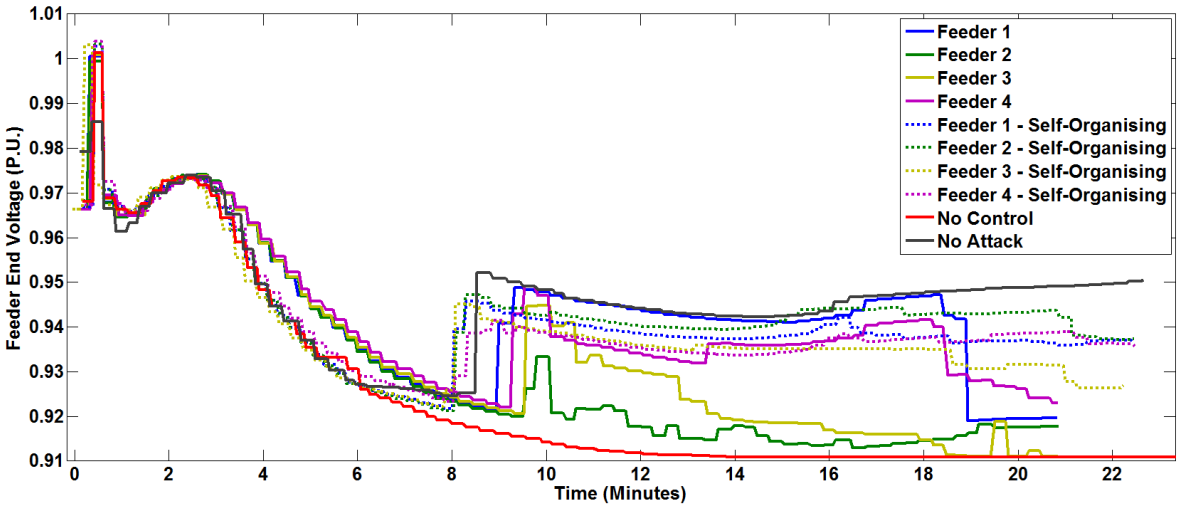


Fig. 8.15 – Voltage Profile Comparison – Adaptive Attack

The figure presents an adaptive denial of service attack featuring 18% customer involvement across the architecture, and illustrates that even under the presence of a more sophisticated attack the self-organising architecture reduces the control deterioration. As a feeder end customers under the self-organising architecture experienced a 1.22% increase in average voltage, and a reduction in the deviation duration by 408.3 seconds.

The following figure in Fig. 8.16, illustrates the computational burden indicator and its components measured across the duration of the simulation. The data was extracted from the simulation using the static denial of service attack. The figure illustrates that the self-organising architecture was able to reduce each of the computational burden components and therefore by extension the overall indicator – which in turn resulted in the ability to prevent control deterioration. The congestion and reactivity components were the strongest contributors to the computational burden in the static architecture, and the self-organising approach achieved a 97% reduction in reactivity and 99.6% in congestion. These significant reductions were created through preventing the message queue from building by increasing aggregation and further distributing customer connections.

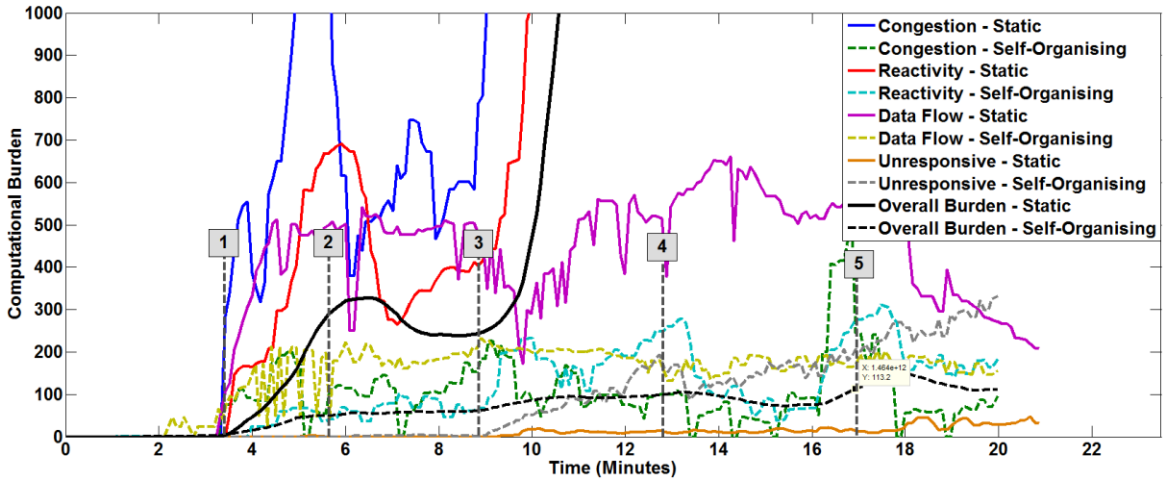


Fig. 8.16 – Computational Burden Composition and Comparison - Static Attack

In terms of managing the data flow, that distribution of connections reduced the data flow burden component by 59.5% - overall this created a total computational burden indicator reduction of 98.4% over the length of the simulation. The figure also presents the timing of the transition events which were triggered by the architect which were as follows:

1) **Rebalancing Action (Stage 1 Transition):** Initially the architect performed a rebalancing action, whereby 10 customer agents were relocated from their current aggregate to a less heavily loaded alternative. This action was the traditional first step and was applied as a pre-emptive decision at the first sign of burden rise.

2) **Activating a Single Dormant (Stage 2 Transition):** The second action was to perform a stage two transition, the decision making engine recommended performing a substitution. However the architect determined that the error reports could not be traced to a single aggregate agent, therefore a substitution would not be effective. Alternatively the architect elected to activate a single dormant agent rather than to follow the recommendation.

3) **Activating Dormant Population (Stage 3 Transition):** The third decision reached by the architect was to activate all of the remaining dormant agents and add them to the active population. In this instance the recommendation from the fuzzy engine is carried as dormant agents were available.

4) **Promoting Agents (Stage 3 Transition):** A fourth transition recommended activating the dormant population – but as those agents were activated in the previous step the recommendation is over-ruled. As no dormant agents are available, the architect promoted customer agents to perform aggregation duties.

5) **Tiered Promotion (Stage 4 Transition):** The final transition performed by the architect was the most severe transition available, involving creating a multi-tiered aggregation layer. An upper tier is formed through promoting existing aggregates up into the new upper tier, while those promoted aggregates are replaced with members of the customer population who are also promoted into the lower aggregate tier.

The following figure presented in Fig. 8.17, examines the performance monitoring data extracted from the aggregation layer with respect to message congestion.

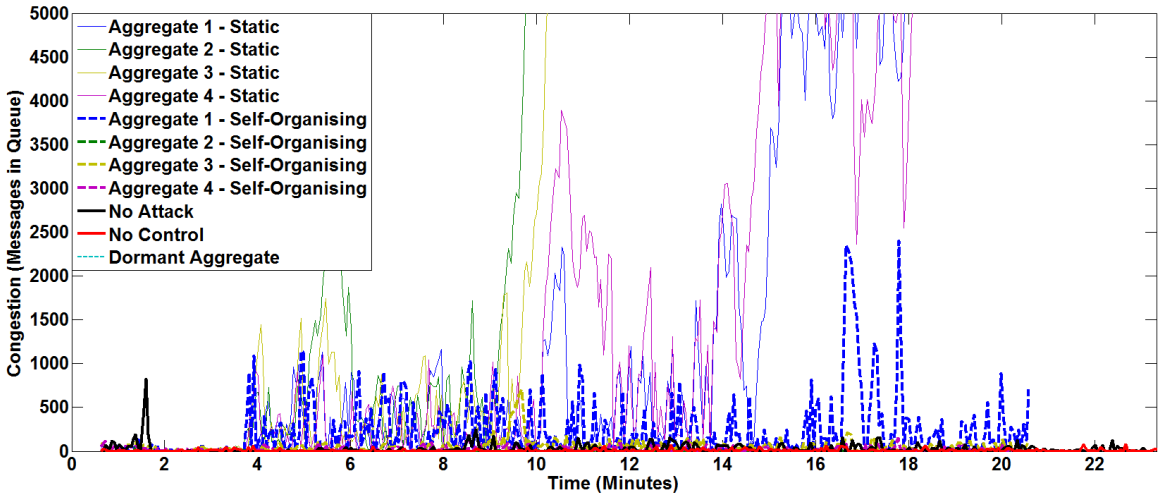


Fig. 8.17 – Aggregate Congestion - Static Attack

The figure illustrates that the congestion data recovered from aggregates operating in a static architecture demonstrates that the message queues are considerably more heavily loaded that those within the static architecture. In the static architecture aggregates 2 and 3 which were responsible for the two feeders with the highest degree of control deterioration experience a sharp rise in congestion earlier in the simulation. This congestion rise 10 minutes into the simulation corresponds with the control loss, additionally when the second two aggregates experience a similar congestion rise after 14 minutes, the control on those feeders also started to falter. This confirmed that the ability to manage the volume of data created by the denial of service attack has a definite impact on the ability to provide voltage control. In contrast the self-organising architecture does not contain aggregates with severe congestion issues and therefore communication between customer and controller was not compromised, and the control objective was achieved.

A similar congestion profile was produced in response to the adaptive denial of service attack as presented in the following figure: Fig. 8.18. The figure illustrates that the more

sophisticated attack proved to be more challenging from a self-organising perspective. Aggregate 3 experiences two periods of raised congestion during the simulation, these periods were due to the nature of the adaptive attack.

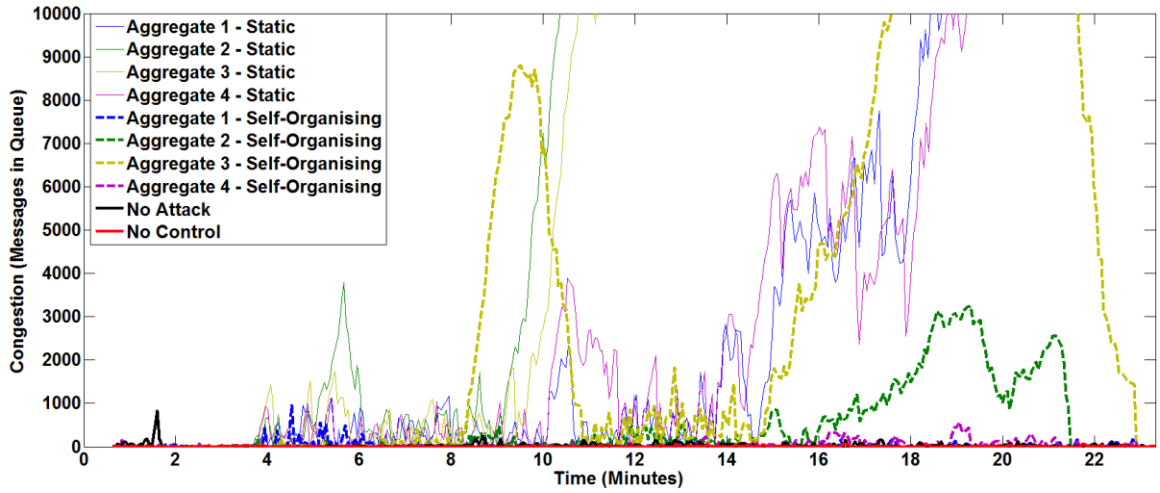


Fig. 8.18 – Aggregate Level Congestion - Adaptive Attack

As an attacker was moved from one location to another, so was the volume of attack traffic it was producing. Therefore an aggregate can find itself under attack from a larger number of attackers during the course of the simulation. This rise in the number of attackers created the rise in congestion, and therefore required a further transition to alleviate. However it illustrated that the decision making engine within the architect agent had the capability to recognise that a transition had created an issue and corrected the problem.

8.3.4 Continuous attack with 24% Customer Involvement

As a final escalation the number of attackers was raised to 80, which amounted to 24% of the total customer population, the voltage profiles for the static variant of the attack is presented in Fig. 8.19.

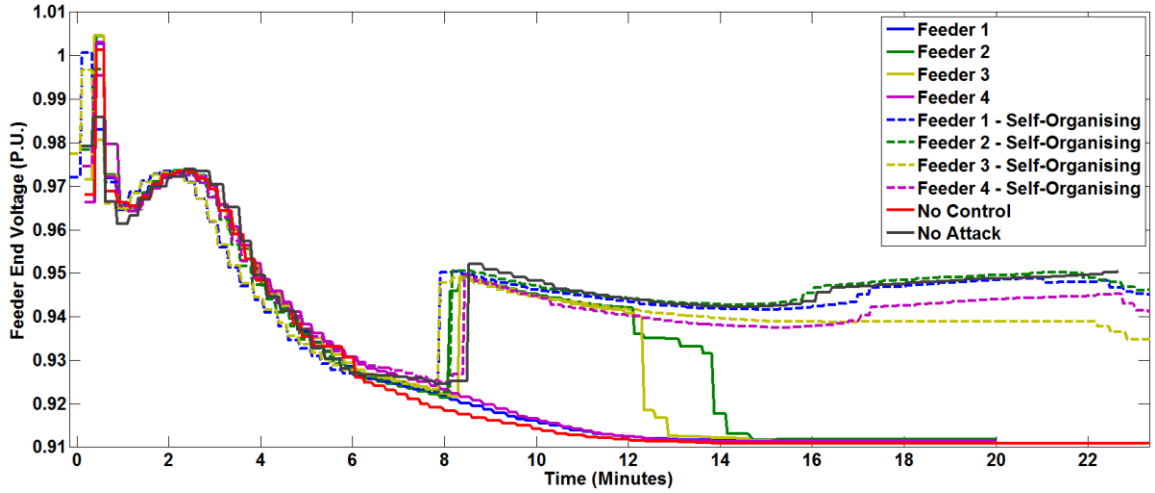


Fig. 8.19 – Voltage Profile Comparison - Static Attack

However in both case of both the adaptive and the static attacks, the self-organising architecture was able to prevent control loss. The figure does illustrate that the voltage profiles representing feeders on the self-organising architecture were lower by up to 0.5%. This indicated that denial of service attack did have an influence on the control performance of the self-organising architecture but this was minimal in comparison to the damage done to the static architecture.

The following figure presented in Fig. 8.20 presents the computational burden data from the static attack.

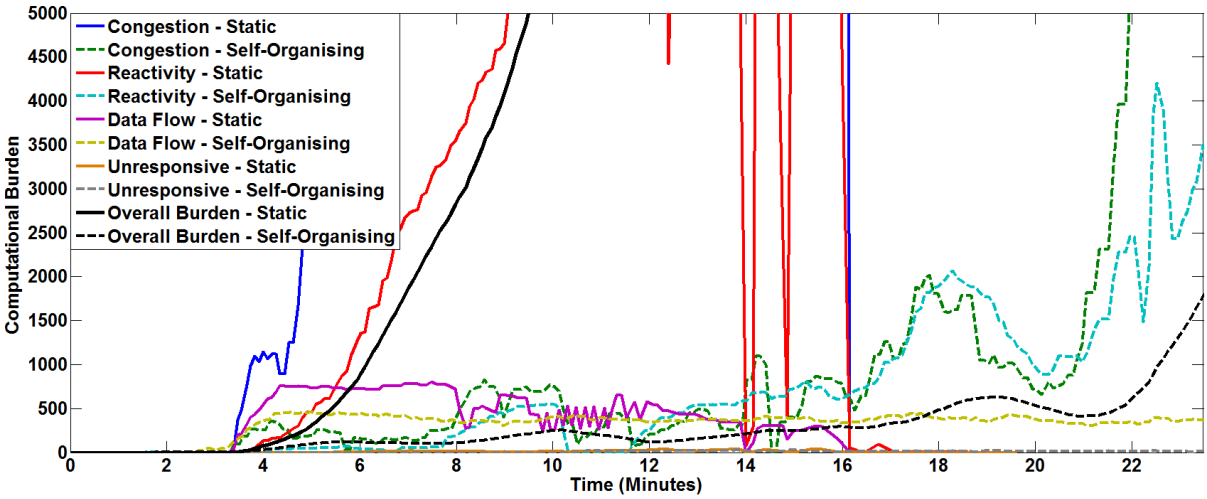


Fig. 8.20 – Computational Burden Composition and Comparison

The figure illustrates the burden information and illustrates a secondary impact of the more severe attack event. In addition to preventing the transmission of control signals and therefore causing control deterioration, performance monitoring messages are also compromised by the denial of service event. Agents within the architecture are no longer

able to transmit data to the architect agent, which created the gaps in the data present for the reactivity burden component starting at 14 minutes, all performance monitoring data is lost at 16 minutes for the static architecture. In contrast the self-organising architecture was able to reduce all of the burden elements – congestion was reduced by 95%, reactivity by 81% and the overall burden index observed a 94.4% reduction. This management of the communication variables ensured that the control objectives were still achieved, and that performance monitoring data could be accessed throughout the course of the simulation. Consequently using self-organisation also prevents a denial of service attack from blinding the performance monitoring aspects of the system.

While the performance monitoring data was not received by the architect agent, the core metrics were retained by the individual agents within the agent population, the following figure presented in Fig. 8.21 illustrates the response times between customer and controller during the static denial of service attack event.

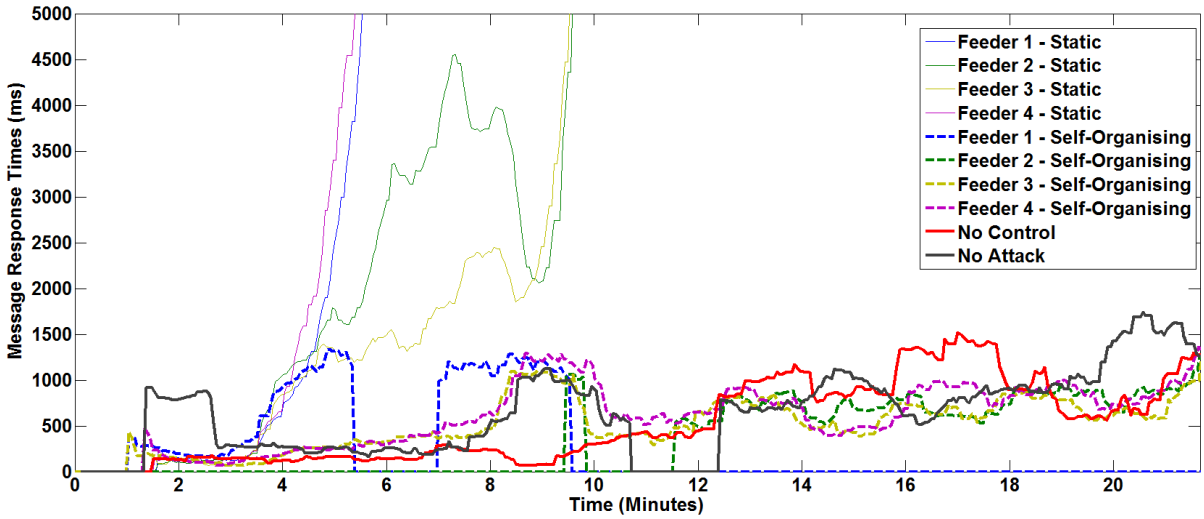


Fig. 8.21 – Response Times between Customer and Controller

The figure details that under the static architecture the feeders experience a sharp rise in response times which equated to the reactivity of the connection prior to observing control deterioration. This further confirmed that the communication performance of the architecture has a link to the system's ability to perform the control objective. In contrast the self-organising architecture does not experience the same rise in response times and therefore the connectivity between customer and controller is stronger throughout the attack. As a result the architecture is able to process and successfully respond to control requests and control queries which prevented the control deterioration observed by the static architecture.

8.3.5 Sequential Attack with 24% Customer Involvement

In addition to the continuous attack formats a further variant of the denial of service attack was the sequential attack. This format consisted of a series of burst attacks during the simulation, each one launched by a total of 80 customer agents and targeting each member of the active aggregate population. Both static and adaptive versions of the attack were performed using the sequential format; the following figure presented in Fig. 8.22 illustrates the voltage profiles extracted from the static variant of the attack.

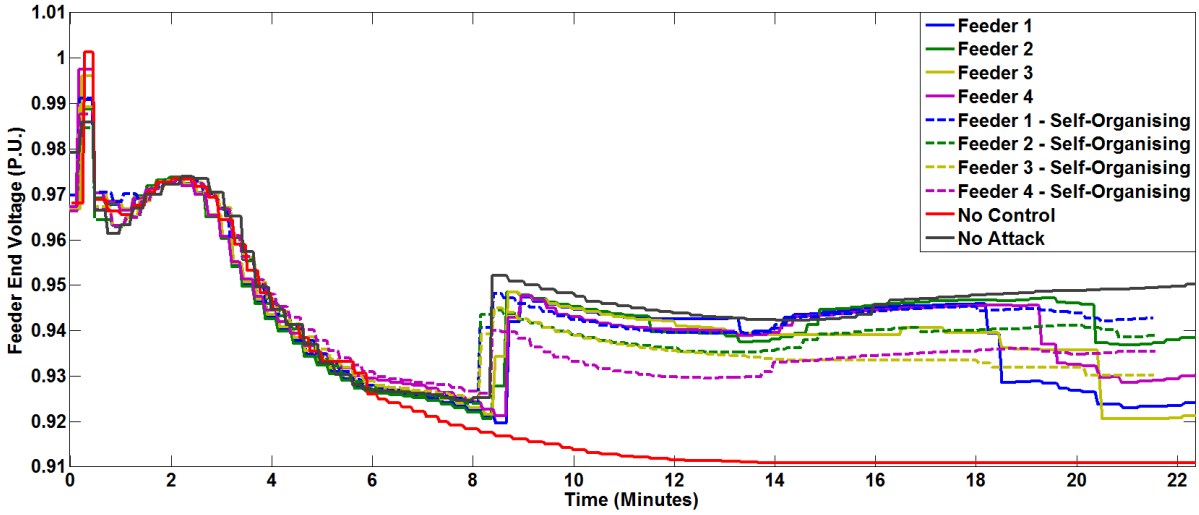


Fig. 8.22 – Voltage Comparison Profile - Static Attack

The results that a sequential attack had a less disruptive impact on the control performance than the continuous attack of the same magnitude, this was due to the intermittent nature of the attack. Between the individual bursts of attack traffic, the aggregation layer was able to recover by processing the message queue before the next wave of the attack arrived. In the static architecture, the final of the three attack events did trigger control deterioration towards the end of the simulation. In contrast the self-organising architecture did not observe any control deterioration during the course of the simulation but the control response delivered by the architecture was initially weaker. Voltage magnitudes after the control were 1.2% lower than those achieved without an attack present. The following figure in Fig. 8.23 presents data from the adaptive version of the sequential attack.

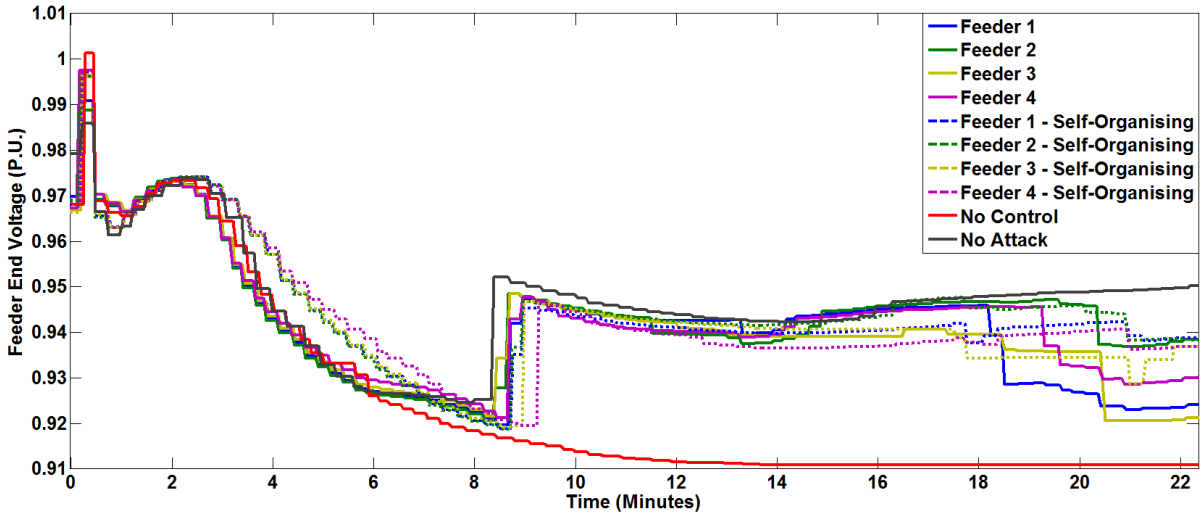


Fig. 8.23 – Voltage Profile Comparison - Adaptive Attack

The figure demonstrates that under more sophisticated attack format, the self-organising architecture observed a small degree of control loss towards the end of the simulation under the final wave of attack traffic. Additionally the initial control response was stronger under the adaptive attack than it was under the static attack. The following figure in Fig. 8.24 illustrates the computational burden data for the adaptive attack.

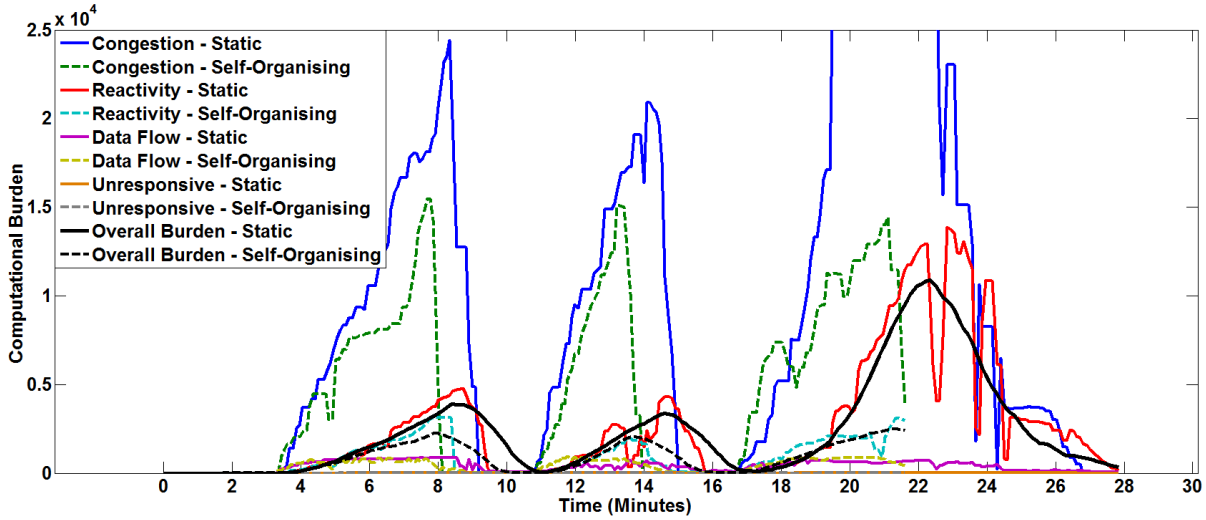


Fig. 8.24 – Computational Burden Composition and Comparison

The figure clearly illustrates the three burst events of the sequential attack, and the third of the three bursts had a stronger impact on the burden indicator from the perspective of the static architecture. As demonstrated in previous examples the congestion component of the burden was the dominant feature and the control deterioration took place when the congestion rose significantly. The self-organising architecture was able to reduce the message congestion within the agent population by 61%, which was a smaller reduction in comparison with the continuous attack events. The overall computational burden was

reduced by 64% and lowered each of the burden peaks during the individual bursts. Another observation is that the self-organising architecture stabilised the burden components across all three stages of the sequential attacks. In contrast the final burst event of the sequential attack did present with a more severe impact on the communication layer – the peak overall burden indicator was 3.2 times larger than the previous peak.

8.3.6 Extended Runtime

A final example considers the operations of the self-organising architecture over a longer time frame to demonstrate that even in the presence of a longer term denial of service event the self-organising architecture was capable of monitoring the deviation through to resolution. The following figure presented in Fig. 8.25 presents the voltage profiles recorded over 50 minutes of runtime with 18% customer involvement.

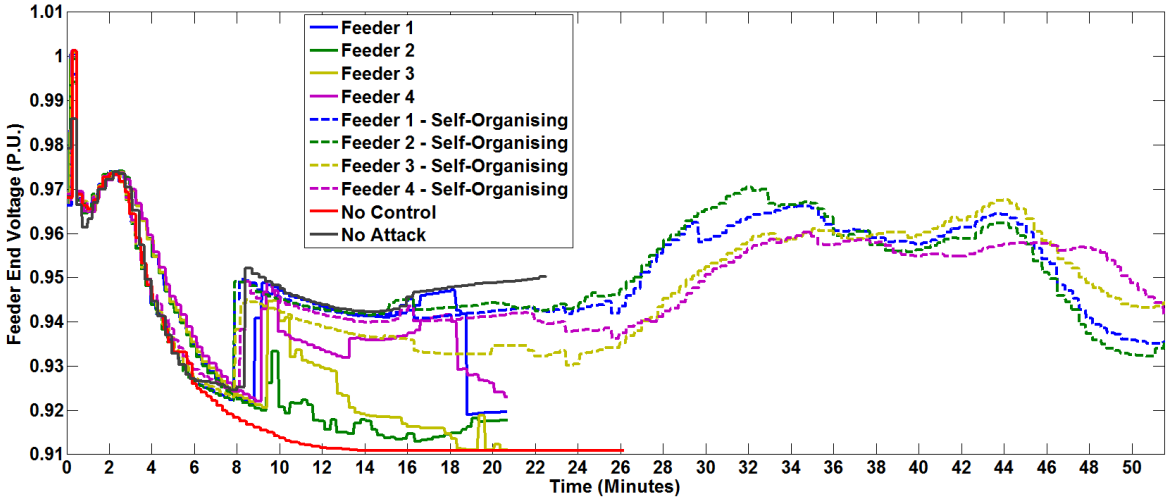


Fig. 8.25 – Voltage Profile Comparison

The figure compares the voltage profiles from the lengthened simulation for the self-organising architecture against the profiles from the 20 minutes simulation regarding the static architecture. The results show that the self-organising architecture did not experience control deterioration throughout entirety of the deviation and restrictions applied to the customer layer were released between 28 and 33 minutes into the simulation. The presence of the attack event did delay the signals to release those restrictions in the case of the second feeder but the control objective was not affected. This demonstrated that the self-organising architecture was capable operating under the presence of an attack event for a longer period of time. An attack population of 24% was not completed for the extended runtime because the volume of messages accumulated by the aggregation layer created instability in the simulation environment. This was more significant when operating the static architecture

where the ability to manage the attack was reduced. The following figure in Fig. 8.26 illustrates the data flow at the aggregation layer during the length of the simulation

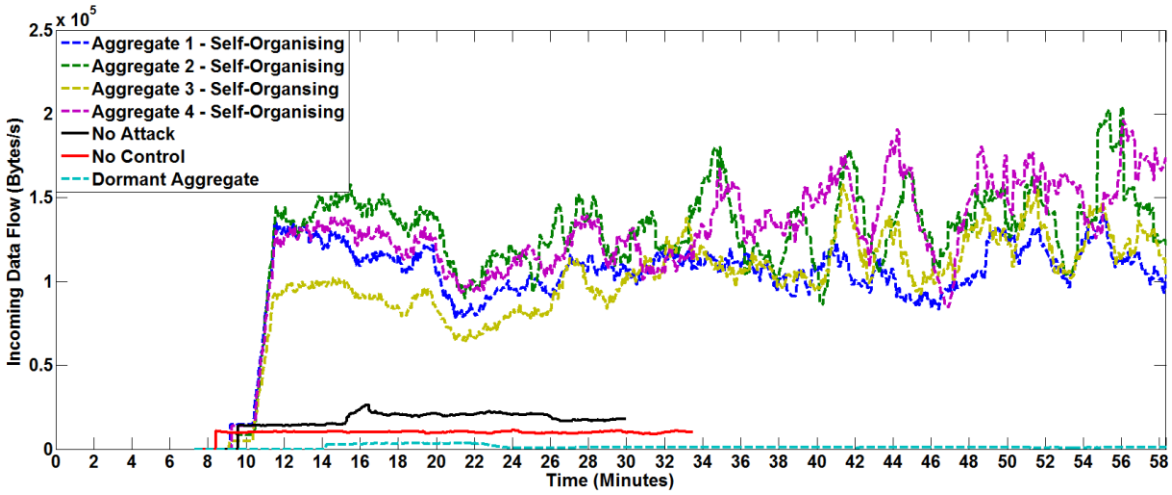


Fig. 8.26 – Data Flow at the Aggregate Layer

The figure illustrates that the attack event was continually delivering attack traffic to the aggregation layer throughout the extended lifetime of the simulation, which explained the delays in sending signals informing customers that load shedding was no longer in place. The potential for the attack to influence control performance elapsed after 32 minutes when no further deviation events were present. However goal of the architect agent is deliver an architecture which manages the computational burden indicator as derived from communication performance monitoring. Therefore in the absence of a control objective the architect is still responsible for managing other properties such as congestion and reactivity. The following figure presented in Fig. 8.27 presents the computational burden components during the lifetime of the simulation.

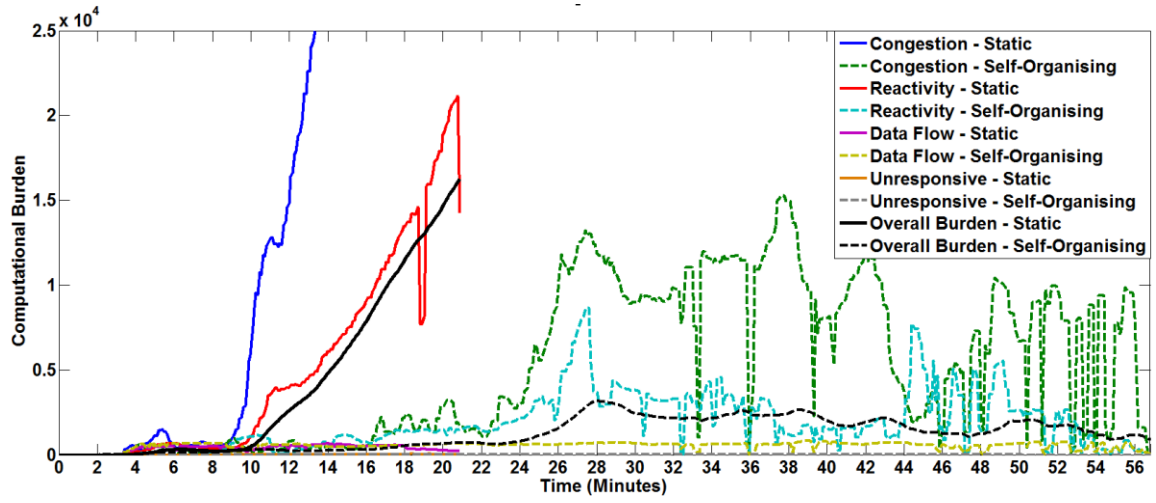


Fig. 8.27 – Computational Burden Composition and Comparison

This figure illustrates that the computational burden recorded by the static architecture does increase during the later stages of the simulation, more so when the control functions have elapsed. However the peaks observed by the self-organising architecture were considerably lower than the congestion and reactivity data from the static architecture observed in the shorter runtime. As a result the overall burden indicator peaked with a value 4.9 times smaller under the self-organising architecture than the static equivalent. On the basis of previous examples, the ability to minimise the burden indicator, with specific reference of congestion and reactivity is important in reducing the likelihood of control performance deterioration.

8.4 CONCLUSIONS

An overall assessment of the results indicated that the self-organising architecture does deliver several improvements over using a static architecture. These improvements are initially delivered through the initialisation stage of the self-organising architecture. Because this process involved customer agents ranking the connectivity to each of the potential aggregates before making a connection, the resulting greater distribution of connections. Customers on the same feeder may be connected to several different aggregates, therefore any clustering of agents transmitting larger quantities of data are broken up increasing diversity. This was represented the results with no attack present and a denial of service event with 6% customer involvement, where control loss was observed using the static architecture in addition to corresponding congestion and reactivity issues. Similar issues were not present in the case of the self-organising architecture and therefore control was not affected, the fact that no significant reconfiguration actions were triggered in those instances implied that the initialisation stage was the key factor.

Attacks consisting 12% of the customer population or lower were not significantly disruptive to the control procedure, but as previously indicated the initialisation stage acted as a first line of defence and formed a communication structure within which the control process was more resilient to attack. In addition to distributing customers' data connections between aggregates the control connections were also distributed. Therefore disabling one or two controllers would not have a detrimental impact on the ability to perform voltage control, because deviations on one feeder could be responded to by multiple controllers.

In addition to the influence of the initialisation phase the self-organising architecture also prevented control deterioration in the event of a more severe denial of service attack. Once 18% of the customer population were transmitting attack traffic the control performance in terms of the static architecture suffered. The average voltage at the end of the feeder by 1.6% to 0.932 per unit in comparison to a configuration without an attack event, whereas the self-organising architecture was able sustain an average voltage of 0.945 per unit. When the attack population was raise to 24%, the denial of service event disabled control across 50% of the network and the remaining feeders lost controllability within six minutes of the initial control decision. This was not experienced by the self-organising architecture as the use of appropriate architecture transitions reduced the computational burden indicator value and therefore ensured that all communications relating to performing control were completed.

This demonstrated that there was a connection between the performance of the communication architecture specifically in terms of congestion and reactivity, and the ability of the architecture to perform the control action. In each of the cases where control deterioration was observed, the computational burden data indicated a corresponding rise in reactivity and congestion. The influence of these particular burden components was that they prevented key messages from being processed at the aggregation tier, and therefore customers did not receive the instruction to reduce demand in the case of the most severe of the trialled attack formats. As the denial of service attack was directed at the communication aspect of the agent architecture, the nature of the control algorithm would have limited influence over the ability to avoid control deterioration. This is because at the point where the algorithm computes the set points for the controllable components – a message is required to apply those settings. A highly congested network would struggle to deliver the command and as illustrated in the example with 24% customer involvement – control was effectively disabled. In instances where the control had deteriorated significantly the congestion at the control layer had surpassed 10,000 messages waiting to be processed, the majority of these messages were generated by the attackers.

As the self-organising architecture demonstrated an ability to reduce the computational burden by over 90% in most cases as a result of preventing large quantities of messages from building at the aggregation layer it indicated that the system would have performance advantages beyond voltage control. It is envisaged that as the smart grid concept gains more momentum and an increasing number of smart-devices are added to the network, the

demand on the communication network is also set to increase. Therefore leading to a situation where core communication requirements for monitoring and control could result in a more congested architecture. This would be further exacerbated through the need to perform multiple control objectives at the same time. The ability to manage and reduce congestion through self-organisation would be applicable even in the absence of an attack. Furthermore the properties of the self-organising architecture indicate that it would be suitable in the management of wider control objectives, for example problems involving protection devices or frequency response require resolution in a shorter timeframe than voltage deviations. Therefore being able to ensure that the communication network is capable of responding in a timely fashion becomes more important.

Overall it can be concluded that the developed self-organising architecture and decision making engine is both functional and successful. It was able to improve the ability to perform the control objective in the presence of denial of service attacks involving up to 24% of the customer population. Additionally it was able to reduce the communicative load on the controller layer which prevented control deterioration and illustrated further potential applications for the self-organising architecture.

Chapter 9: Discussion

9.1 INTRODUCTION

This thesis describes the work undertaken to investigate the relevance of self-organising architectures and the subsequent development of a proof concept implementation of such a system. In this chapter the potential applicability of such a system is presented, within the current power system landscape and the system requirements involved in translating a proof of concept development into a real-world roll out scheme. Furthermore this chapter outlines additional features which could be included in further developments of the architecture and the decision making approaches at the heart of the architect agent. Finally this discussion chapter offers avenues of further research using the core self-organising architecture as developed over the course of the documented research.

9.2 DISCUSSION

Upon review of the final result set and the overall design and development process of the self-organising architecture, several points for discussion were raised from the performance of the individual stages of operation to the finer points of detail involved with facilitating those stages. The following section aims to explore those points of discussion and consider their influence of the results which have been presented.

9.2.1 Performance Overview

The results suggest that there is value in the self-organising architecture with respect to reducing the computational burden. Lowering the burden had positive results on performing the voltage control process, as a result of the controllers being under less communicative load and can therefore respond to control signals and queries from the customer layer of the hierarchy. The first observation is that the attack format delivered through a series of noise messages transmitted from compromised smart-meters to the control layer experienced consistent control degradation under attacks launched by over 12% of the customer population. Therefore under denial of service attacks with fewer attackers, the voltage control remained largely unaffected even through the transitional decisions performed by the architect in response to the performance monitoring information did successfully reduce the computational burden indicator by 65.6% and controller congestion by 93.1%. Therefore indicating that in addition to improving control resilience, the self-organising architecture has value in handling heavily loaded communication architectures. As the penetration of monitoring devices and controllable components increases within the smart-grid environment will be responsible for more challenging control situations

featuring multiple objectives. Reducing congestion and controller response times in such a scenario will become more important even in the absence of an attack event and therefore find value in the application of a self-organising architecture. Furthermore the self-organising architecture improved control response times with a small scale attack involving 6% of customers by 67.9% equating to an improvement of 3.1s on average. Therefore indicating that the self-organising architecture could also facilitate alternative control objectives such as frequency response or protection, where smaller scale attacks would otherwise prove disruptive. The reduction in computational burden delivered through the self-organising system may also be applicable to protection systems – which if the system is under attack from an adversary intending to cause blackouts or overload physical components, being able to deliver protection signals becomes even more important. This indicates that while the architect agent was focussed on reducing computational burden as this formed the input to the decision making engine, the results have a positive impact on influencing the electrical properties of the power system.

9.2.2 Decision Making Engine

The core component of the self-organising architecture was the decision making engine, which was responsible for translating performance monitoring data into architecture transitions. This decision making engine took advantage fuzzy techniques to handle the level of uncertainty involved in the calculation of the computational burden indicator, which proved to be an appropriate mechanism for evaluating the error state of the architecture. The challenge of implementing the fuzzy system involved converting the data from the different performance metrics, each of which featured differing units and measurement criteria. Therefore each error event was normalised in reference to its threshold value, converting it into a severity percentage. Each recorded error report needed to be combined with respect to the tier of the architecture the error arose from using the severity percentages. Because each of the severities related to a different threshold in respect to each of the metrics the burden indicator remained a dimensionless quantity. Configuration tests were performed to evaluate which levels of computational burden constituted a low, medium or high event and therefore used to calibrate the fuzzy input membership functions.

The output of the fuzzy decision making engine took the form of a recommendation for a transition. However the architect agent in which the decision making engine was hosted

had the capability for over-riding the recommendation. This would be the case in the event that the recommendation was not feasible given the available resources, for example denying the activation of dormant aggregates if all available aggregates were active. This procedure acted as a functional layer of transition validation before any actions were triggered, in each of the simulations, the architect was able to avoid performing impossible transitions. The need for the ability to over-ride decisions indicates that a fuzzy decision machine engine requires additional support with respect to the location and distribution of errors, and the available resources.

A combination of these properties were proven to be effective as the results demonstrated the decision making engine was able to use the burden indicator to deliver effective and appropriate architecture transitions. Across the range of attack magnitudes and considering both static and adaptive denial of service strategies the computational burden indicator was reduced by 89.6%. Furthermore the decision making system was also able to raise the average voltage by 1.5% when sustaining the most severe attack format triggered by 80 customer agents. Therefore it can be concluded that the selected decision making system functioned correctly was capable of facilitating suitable architecture reconfigurations.

9.2.3 Fuzzy Recommendations

The decision making engine was centred on set of fuzzy membership functions which were responsible for recommending a transition event. The computational burden indicator and its rate of change were supplied as inputs. A fuzzy based system was used as the core element of the decision making engine due to its ability to handle sources of uncertainty. Several sources of uncertainty involved in the decision making engine, in terms of the thresholds for the individual metrics and the computational burden indicator calculation. Therefore it was deemed appropriate to implement a system which was capable of operating under these circumstances.

Alternative approaches to decision making could have also been applied, and prior to the implementation of the fuzzy system, the decision making engine was powered by a decision tree. This approach was capable of converting performance data into transitions but lacked flexibility in terms of incorporating new metrics and relied on definitive decisions without taking into account the sources of uncertainty. In both cases the decision making engine was playing a reactive role in responding the current state of the architecture and applying a transition to reduce the computational load. An alternative approach would be to apply

an optimisation approach which assessed all of the potential transition options and select the most effective structure for the architecture. This approach was not selected for several reasons, the first of which being a lack of a modelling method to validate each of the potential configurations before applying one. In power system optimisation is it possible to use the physical properties of the network and a series of equations to determine precisely what impact a particular change will have on the system state, and therefore being able to exhaustively demonstrate which of the parameter sets is the most effective. When assessing communication data extracted from a live series of interactions, this validation loop is not applicable because of the sources of uncertainties involved. It is not always known what is causing the issues in the communication network and therefore making defining an optimal configuration less feasible. The level of uncertainty is increased when the architecture is under attack from a human adversary, an optimisation mechanism will not be able to devise a solution because it will have no information about what the attacker plans to do next. The issue faced by the self-organising architecture becomes less about optimality and more about maintaining service availability. An attack will not necessarily behave in a deterministic manner and therefore cannot be used as a mathematical input into an optimisation technique. Therefore for the application presented in this thesis, a fuzzy based decision making engine was an appropriate approach.

It can be accepted that the current design approach could be enhanced through the application of a learning and tuning approach to the fuzzy decision making engine whereby the architect agent builds a library of events, error states and decisions made with a view of retraining the set of membership functions going forward. Under this scenario it may be relevant to define and engineer a metric to define effectiveness such that the architect is able to determine and remember whether the selected transition event had the desired impact on reducing burden and improving control performance. These enhancements would add further value to an already proven mechanism and improve the potential for their use in wider applications

9.2.4 Control Selection

Another design feature employed in the research was the control mechanism. The control approach takes advantage of emerging techniques involving the active participation of customers in network management. Therefore a demand side response mechanism was devised, involving reducing customer demand to resolve an under voltage deviation. The

selected control approach required customers to periodically ask the controllers if restrictions were needed. This control format was sufficiently able to correct the voltage deviation, and in the absence of any attack event, the deviation was resolved within 207 seconds of first falling below 0.94 per unit. Consequently the voltage was not significantly low enough for several minutes and therefore the control mechanism can be considered to be effective and successful.

Alternative voltage control mechanisms are available, and may deliver faster resolution times, but this investigation was not a discussion of the overall performance of voltage control algorithms. Furthermore the alternative control approaches will also require the transmission of data from point of measurement to the controller, and in response the controller will need to send control signals to a relevant controllable device. This may take the form of signals could be sent to a transformer requesting a tap change or to an energy storage device to discharge during peak demand. A denial of service attack directed at the controller delivering these signals would experience the same challenges involving message congestion and therefore message reactivity. Consequently the control algorithm would be under the same pressures as those documented in this research, and therefore the selected control method is valid. An additional consideration is that alternative control signals may require the interaction of a larger variety of devices and increase the complexity of the communication traffic. As the self-organising architecture has proven it was capable of improving communication flow through reducing congestion and response times, more value would be gained through implementing such a system in the presence of an alternative control algorithm.

9.2.5 Agent Platform

The self-organising architecture was built upon a set of agents using the JADE agent platform, this platform was used due to its ease of use and repeated usage within the research community. The agent platform provided tools for examining the inter-agent communications which functioned as a troubleshooting tool during the development process, and ensured that the architecture was functioning correctly. Some limitations were present with the use of java based platform in terms of ultimate scalability as a result of the agent platform taking up space within the java virtual machine. As architectures with less complex agents can accommodate a larger population, the additional overheads involved with providing self-organisation reduced the effective agent capacity of the platform.

Other agent based platforms are available such as presage2, and have been applied to the smart-grid domain. The use of an alternative platform may respond differently to the attack events and experience different levels of congestion, but the self-organising techniques discussed and developed in this thesis would remain applicable. The aim of the research was not focussed on defining and engineering the most effective agent based control and monitoring system. Instead the core objective was the investigation and subsequent development of a self-organising architecture for the purposes of providing resilience to cyber-attacks.

An alternative approach – more applicable potentially for a real world deployment would build an agent based architecture without the overhead of the over-arching platform. Instead agent communication would be communicated via socket connections between components with additional encryption and security requirements and a communication protocol in line with the existing hardware. Such a format may avoid limitations involved with scalability on the behalf of the agent management system when running complex agents when considering much wider agent populations. However as a result of not having the agent management system, tasks such as name resolution, directory services and communication processes would have to be developed in replacement of the functions provided by the agent platform.

9.2.6 Attack Format

Following the development of the operating components of the self-organising architecture a further consideration was to the design and implementation of the attacks launched against the system. The selected approach was based on a low-rate denial of service attack, targeted at the controller layer. Because this was a novel and emerging research area, no previous research had considered the use of self-organisation for the purposes of defending against a denial of service attack, no standard model existed. The denial of service attack was selected due to it targeting the network layer of the supporting ICT infrastructure, therefore influencing inter-agent communications. The aim of the attack was to disrupt the flow of information between customer agents and the control layer located at the aggregation tier, and therefore trigger control deterioration. Furthermore a denial of service attack was one of the core components involved in the Stuxnet and Ukrainian events, thus demonstrating its relevance when discussing vulnerabilities in cyber-physical power systems.

As set of design choices were made in the development of the attack strategy including the degree with which the attack was modelled. In the research the attack originated from compromised smart-meters – because these devices are readily accessible in customer homes and could be connected to other smart devices over an internet connection. This therefore made them reasonable targets from an attacker's perspective. Given the significant number of potential permutations for number of attackers, distribution of attackers, attack duration, severity and sophistication, performing an exhaustive examination of all possible formats was not feasible. Therefore a continuous attack approach was the core focus of the investigation as this represented the longest duration for each of the compromised customer populations and was supported by additional examinations of burst and sequential attacks. A secondary decision choice relating to the use of smart-meters at the launch platform for the attack was the volume of attack traffic. The designed volume of data transmitted from each compromised smart-meter was configured such that it fell in line with the transmission capabilities of the smart-meters designed for the current UK rollout. Therefore the strength of the denial of service attack was limited by the properties of the attacking hardware; an adversary delivering attack traffic from an external source would be able to deliver a more intensive attack volume.

Alternative attack mechanisms such as false data injection or a man-in-the-middle attack may require additional functions and an alternative approach from the self-organising architecture. This is because these attack methods do not necessarily impact on the communication volumes and data evaluated by the performance metrics devised in this thesis. The onus would be on data verification and restructuring the architecture on the account of a trust vector applied to those agents delivering false information. Such attack methods would be the domain of future work which would add further value to the results illustrated in this thesis.

9.2.7 Scalability

As described earlier in this thesis, scalability is one of the core properties of a self-organising system allowing them to accommodate large numbers of components. Therefore part of the initial development and research presented in chapter thee, scalability was one of the metrics used to differentiate between individual architecture topologies and assess relative performance. Each of the architecture designs was evaluated with three differing customer agent populations of 540, 1080 and 1640 agents in addition to generation and

aggregation agents in supporting roles. The investigations demonstrated that the scalability of an architecture was related to its design and the manner in which the communication between agents was managed, increasing aggregation improved scalability. This research also determined that a configuration without dedicated aggregation agents struggled considerable in managing data flow when the number of agents was increased. Therefore such a configuration was not considered for implementation in the final self-organising architecture as an option which the Architect agent could select as part of a transition mechanism. Using preliminary research as the input and foundation for the development of the self-organising architecture allowed for scalability criteria to form part of the design process and eliminate known sources of scalability issues early on. The disaggregated architecture was one a known source of scalability issues because generation agents were required to accept the responsibilities of performing aggregation in addition to their core objectives, as the number of agents increased so did the congestion at those agents resulting in longer response times and reduced controllability.

Within the developed self-organising architecture itself the scalability of the system can be influenced by several factors including the number agents involved and the communication structure those agent form. But the simulation environment can play an equally influential role in determining the scalability of the architecture, while the multi-agent approach is composed of numerous individual elements interacting with one another – the entire population is overseen by a single agent platform hosted on a local machine. The design choice to utilise a single machine was motivated by the impracticalities involved in developing agents on hundreds of individual devices, which would have become even more impractical when considering populations over 1,600 agents as examined in the preliminary research in chapter three. In terms of a potential physical deployment of the self-organising architecture the individual agents could easily be hosted on small devices such as Arduino, mBed or Raspberry Pi style platforms and therefore locally hosted environments would not create a scalability issue.

The simulation environment is driven by a java virtual machine which is reliant on its own allocation of virtual memory in the form of heap space, each agent active within the simulation environment occupies space in this virtual machine and within the heap space. A larger agent population therefore requires a larger proportion of the available heap space and therefore each agent has less headroom in which to operate. Due to the increased functional requirements of the agents involved in the self-organising architecture the upper

limit on sustainable agent populations within the virtual machine was reduced, hence simulations were not conducted to the same magnitude of those in chapter three. A further aspect of heap space management relates to messaging between agents, each transmitted message exists within the virtual memory until it has been processed by the recipient at which time it is handled by the java garbage collection process. If the virtual memory requirements of the simulation exceed the maximum available heap space, the simulation is automatically terminated – which in turn limits maximum runtime. As more congested architectures will consume more virtual memory, their maximum runtime is shortened in comparison to an architecture with fewer unread messages. Therefore a system which is capable of minimising the congestion within the architecture would create benefits for scalability and the self-organising architecture presented in this research improves the scalability potential in comparison to a static version of the same agent population.

As computational limitations of the host machine were the core source of potential risks to scalability concerns it can be determined that the set of agents as a whole demonstrate scalable properties. Each of the stages of the self-organising architecture implement adapted versions of generic techniques which in their native research domains have been claimed to be scalable, and the methodologies could be translated into a variety of platforms beyond the agent based approach. For example in the case of an infrastructure of independent sensors, the same processes and functions can be applied but without the limitations of a single local system. The central agents in the system – Observer, Architect, and Gateway – would be located within a central resource which would have sufficient computational power to process initialisation states, performance monitoring data and reconfiguration interactions.

9.2.8 Evaluation Approach

Examining the evaluation method takes into account the platforms and simulation environment applied to both the development stage and the performance testing format. All the simulations were completed on a single platform hosting the agent population and the supporting load flow calculations within Matpower. This process was selected on account of the available resources and the tools used to develop the self-organising architecture. With 365 agents involved in the self-organising architecture it was not practical to engineer a system with each agent running on separate hardware communicating over a network. Some of the agents could have been implanted on different machinery but a decision choice

was made to deliver an even playing field for each of the agents and therefore minimising the number of external influences.

Alternative approaches may involve introducing physical components and additional simulation tools in a lab based environment to further research. In this initial proof-of-concept stage it was relevant to remove external influences or points of failure from the testing regime such that the strongest influences over the performance were down to the interaction between members of the agent population and their response to triggered attack events. Introducing physical components would require additional interfacing work and would be seen as a further step now that the self-organising architecture has been proven in simulation.

For example introducing a real-time power-systems simulation environment may encounter difficulties when communicating with the java based agent platform. Several agents would require being redesigned to match the capabilities and settings of the simulation platform. In this thesis it was discussed how the agent platform interacted with a static load flow engine provided through Matpower. However those requirements would change when operating with a real-time simulation tool, as different elements of the simulation may be handed over to the power-system tool rather than being modelled within the agent population itself. For example load and generation profiles would be the concern of the power system model rather than being rendered by the individual agents. Further work in co-simulation or real-time simulation would be advantageous and would add value to the work completed in this thesis but was not the subject of the presented research.

9.3 RESEARCH APPLICATIONS

The end product devised based on the preceding research had been developed on the basis of inspiration from concepts and methods drawn from a range of domains and tested using a java based multi-agent platform. Several applications can be drawn from the research both in terms of the power systems domain but also in the wider research community.

9.3.1 Architecture Selection

An initial contribution focuses on the relevance of the architectural design of the control and monitoring infrastructure. Structuring and managing the interaction between agents, data collection points, controllers and administrative agents is beneficial in terms of communication management. Improving communication management then has benefits for

the physical element of a cyber-physical system. Control signals delivered by agents to embedded controllers within the network management environment are ultimately subject to delays and latencies generated by the communication layer and therefore improving the performance of the communication layer has a positive impact on controllability. The early research described in this thesis demonstrated that the design of the monitoring architecture favoured certain properties either in the form of lowering communication congestion, or improving robustness to failure. Therefore in the absence of self-organising architectures in the short term, specific static architectures may be deployed in smart grid projects. In such a scenario the control and communication architecture should become as important a design choice as the electrical infrastructure and deployment of emerging technologies such as energy storage, renewables and power electronics. Both aspects of the cyber-physical system would need to be designed to complement one another, and be designed with each other in mind. For example if a prospective smart grid development is anticipated to incorporate a large amount of volatile distributed generation units – an architecture specifically designed with reactivity in mind would be more appropriate. On the other hand a system focussed on residential smart-metering where pricing updates, DSR signals are on a minute time-scale and are less time sensitive it would be more applicable to structure the architecture around scalability and congestion management.

9.3.2 Property Extraction

Secondly the individual properties of the self-organising architecture can be applied as further design choices in support of those previously highlighted. For example the dissemination of customer and aggregate connections formed by the initialisation stage of the self-organising architecture created a more stable foundation for operating the communication network. This is of particular use in respect of building in fault tolerance into the design of the ICT infrastructure without an excess of redundant hardware. The initialisation process creates an overlapping communication mesh whereby control signals for one area of the network are distributed across multiple controllers. As a result the loss of multiple controllers does not necessarily result in the loss of controllability over the network population, additional redundancy can be added through a connection between the controllers and a central server with both physical and ICT network topology information such that any network knowledge lost through failure can be reclaimed by other nodes in the architecture.

The performance monitoring aspect of the self-organising architecture can also have further reaching applications and illustrates the importance of being able to continually measure the condition or usage of ICT assets within the network. The communication and monitoring components of the system may not have the same commercial value of larger power system assets. As a result the replacement of failed components does not carry the same economic penalty, but being able to used monitoring techniques for the purposes of assessing state of health can be beneficial. Components will be communicating measurements of the hardware under observation and additional parameters could be added to that data stream to perform self-diagnostics.

Additional performance criteria would be needed in a real world implementation that document the physical health of the components – temperature and humidity sensing may indicate potential hardware failure. The architect agent in the developed self-organising architecture used these reports to detect predominantly software related issues – message congestion, slow response times. This information could be used in association with a wider spread of metrics as a potential method for predicting component lifespans and informing a maintenance schedule. Self-monitoring data delivered by the components may be interpreted by a human controller, or processed by a decision making engine to add self-monitoring capabilities to a static architecture.

9.3.3 A Self-Organising Architecture

While each of the components of the developed self-organising architecture has value in isolation the system as a whole would be applicable for a smart-grid implementation. Additional work would be required in physical rollout of a self-organising architecture, but the overall concept has been illustrated as being fit for purpose and with demonstrated benefits. Therefore the work in this thesis could be considered as a foundation investigation providing cyber-security benefits through self-organisation. Even in the absence of a cyber-threat being able to monitor for failures within the IT network or indications of pending failure or performance loss could be useful when considering network management systems.

9.4 IMPLEMENTING A SELF-ORGANISING ARCHITECTURE

During this research the system was developed and evaluated through simulation, however in terms of transferring a similar system into a physical rollout there are a number of considerations which would need to be addressed.

9.4.1 Control and Monitoring Hardware/Software

At the core of the system is a series of agents – each representing either a physical component located within the network, or a service provided by a central server or a cloud based solution. Therefore as part of a rollout process it is important to consider the hardware required in providing the monitoring and control capabilities envisaged. The first component is the customer smart-meter, at present smart-meters are in the process of being increasingly prominent in providing usage information to suppliers as part of the smart-meter roadmap. However as a result of the manner, in which the energy market in the UK is configured, the meters are provided by the energy suppliers and therefore neighbouring customers may receive differing models of meter. Therefore a degree of interoperability is required between differing versions of smart-meter, furthermore these meters may require multiple communication channels for separating consumption data transmitted to the supplier from control actions supplied from the network operator.

A second consideration refers to the devices responsible for performing data collection points or controller services. The location, communication capabilities and hardware configurations of these devices is important to consider. The self-organising concept presented in this thesis considered agents which are not physically mobile and can be connected to a permanent power supply and thus removing potential concerns for battery life and energy efficiency. However it is entirely feasible that in a real-world deployment certain agents may not have readily available access to a power supply and therefore are battery operated. In this scenario the acts of the architecture to reduce message congestion and communicative load will in turn have a positive impact on the battery life of those components. A variety of agents with and without power supplies would change the dynamic of the self-organising architecture. Certain agents would be prioritised over others such that those agents running off a consumable power source are recognised as components to manage more efficiently than others.

A final set of agents to consider in a real world implementation are the observer and architect agents, these being centralised entities would either be physically located at a control station on a server or hosted by a cloud computing service. In each case multiple instances of these agents could be hosted on the same hardware are respond to different network zones, for example on central control room may operate several domains. With respect to the gateway agent, its purpose within the simulation was to interact with the Matpower representation of the network and report voltage information back to the agent

population. In a physical implementation it is not necessarily the case that such a stage would be omitted as system modelling may form an important part of the control stage in the form of state estimation. As previously indicated an optimisation approach may not be feasible when processing attack information and issuing architecture transitions, but it does form an important role in power system control. Maintaining access to a system model updated with information from the agent population could be useful in running validation assessments of planned control actions, or for the purposes of running optimisation techniques. The gateway agent could also be co-located with the observer and the architect and collect information from the agent population. Alternatively multiple network models could be maintained at local controllers and therefore each controller will adopt functionality used by the gateway agent.

9.4.2 Communication Infrastructure

Outside of a simulation environment the communication between elements of the architecture have to be considered. At present deployed smart-meters in the UK communicate data through infrastructure managed and monitored by the Data Communications Company [143] and regulated by Ofgem. Therefore the core communication infrastructure is in place and described in the following figure - Fig. 9. - sourced from [143], indicating that the data is delivered through the wide area network through to a set of users. On this basis it could be considered that the ICT infrastructure would be capable of hosting such a system, as bandwidth availability will naturally increase in the future and could accommodate a larger scale deployment of the architecture.

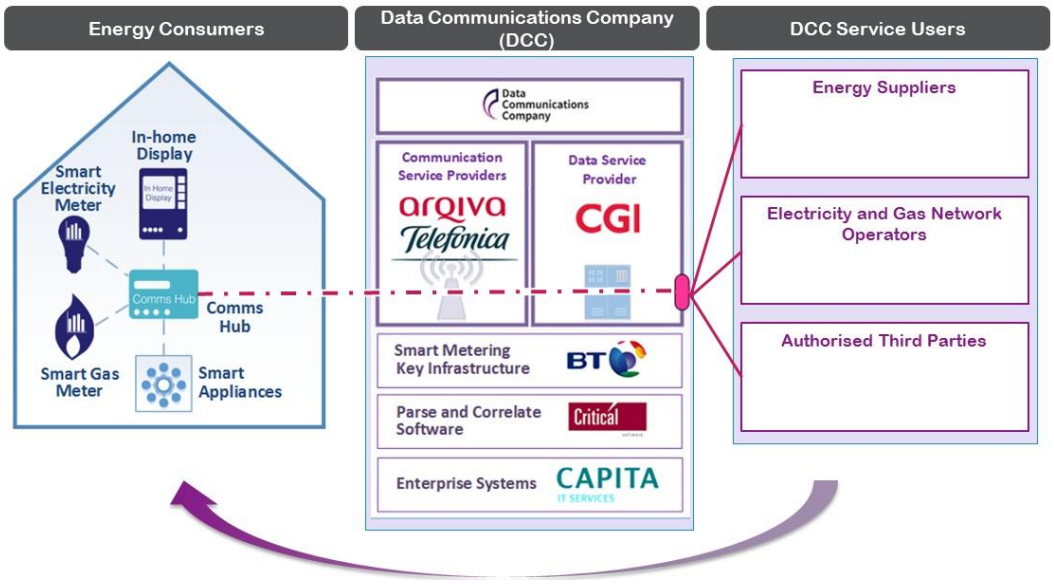


Fig. 9.1 – Current Smart-meter Communication Structure

In the event of the implementation of a self-organising architecture the architect agent would have to utilise this infrastructure to send the relevant messages to the customer layer and to data collection points within the communication layer of the above diagram. The performance monitoring information would likely be received by the network operator as while the self-organising architecture predominantly impacts on the communication architecture, those transitions are made for the benefit of the electrical network.

9.4.3 Security Considerations

All parties involved with utilising the infrastructure are obligated to operate under the smart energy code as defined in [144] with authentication protocols outlined in [145]. One of the primary concerns presented in the documentation is data access, Fig. 9. indicates that other than the network operators and suppliers the data can be transmitted to third party users. Therefore there is an authorisation step which involves user certification and encryption key access to ensure that the process is not being exploited.

Other security concerns are present at the opposite end of the communication structure in the link between the customer smart-meter and the wide area network. Controls will be needed not only to validate that the smart-meter is a registered but also the content being transmitted from it. While it is not be feasible to create a system which is capable of being immune to all forms of cyber threat, defence mechanisms can be used to prevent attacks from adversaries with a limited skill set or with limited resources, therefore reducing the number of attack events which can bypass security procedures.

Finally the integrity of the smart-meters themselves could be vulnerable to malicious use. In the presented research, compromised smart-meters acted as the launch platform for a denial of service attack event. This represented the scenario whereby the firmware of the device had been modified or malware installed which had the capability of sending a wave of attack traffic to the controller. This may be more of an issue if the smart-meter supports the use of open source software [146], while there are numerous benefits to the usage of open source code, there are inherent risks of that code can be modified and exploited within the meter itself. Open source software may extend to third party applications as the smart-meter becomes more central in a home energy management context. The applications would need to be moderated and verified before being released for public consumption to prevent the distribution of malware or exploits. Furthermore the third party applications

would need to be limited to certain levels of access with respect to which information they can view or modify, and communications generated by the app should be restrained within the home area network and not released into the wider communication enviroment.

9.5 DEVELOPMENT POTENTIAL

Over the course of the research period a functioning self-organising architecture was developed which successfully delivered performance improvements across several network management criteria. However there are some potential improvements which could be made if the research was carried forwards, these improvements would further enhance value of the self-organising architecture.

9.5.1 Decision Making

The first development stage would focus on the decision making engine within the architect agent – the version presented in this thesis contains fuzzy logic based recommendations informed by performance monitoring data. This service can be enhanced through the addition greater awareness of the physical properties of the network, for example voltage data could become a further input into the decision making process. Therefore the architect would be more likely to make larger scale transformations if it was aware of a deepening voltage deviation and knew that such a transition would make a positive impact on voltage. While voltage information was recorded and used for control purposes it did not form an input to the decision making engine. This was a design choice which was made because the architect agent could not influence the control algorithm, and was tasked with performing structural reconfigurations within the agent population. Secondly the denial of service attack was launched against the network layer of the architecture and therefore the performance metrics were centred on responding to the cyber-threat. As with the computational burden data the architect would need to be provided with the voltage magnitude at the most severely affected bus and the rate of change of that voltage. To provide such information an updated voltage monitoring mechanism would need to be installed within the customer layer, such that it maintains a record of previous voltage readings in the same manner that the communication metrics are monitored. This would likely be implemented through the provision of an additional voltage control object rather than internal agent behaviour as it was developed over the course of the presented research.

In addition to the use of physical properties as method of informing the decision making processes, it may also be beneficial to provide additional filtering and knowledge to the transitions made as a result of the decision making. The research demonstrated that there were instances where the actions of the architect introduced an aggregate agent into the active population only to create an increase in computational burden rather than reducing it. To avoid such a scenario the architect would need to perform a selection process on the available dormant agents to ensure those that are activated would not become problematic. Locational awareness may also be useful in sculpting transformation events, firstly in terms of activing dormant agents which are in close proximity to the area of the network requiring the additional aggregation capacity. Furthermore this location data could prove useful when selecting substitute agents and structuring the initialisation stage of the process.

Once a decision has been completed a further point of expansion would be to add memory of transition events to the architect agent – such that it can learn from previous decisions and the effective impact they had. Therefore using former knowledge to inform future decisions, or override recommendations from the fuzzy decision making engine – alternatively the machine learning process could redefine the fuzzy membership functions on the basis of previous transition events.

9.5.2 Modularity

A secondary development step would be to introduce greater modularity, as this would improve the ability to evaluate a larger set of scenarios – especially when coupled with an automated testing system as presented in the previous sub-section. A modular approach to the development of the self-organising system improves the ability to perform comparison studies between different networks, control algorithms and decision making methods without having to rebuild a large percentage of the overall system. Furthermore if larger elements of the system remain the same comparison between swapped modules become more valid.

9.5.3 Automation

In addition to improvement the implementation of the functionality of the system, it would also be beneficial to improve the automation of the testing process. Unlike conventional power system simulation techniques where the system states are calculated based on the physical properties and hours of simulation can be completed in seconds, the self-organising architecture operates in real time. The multi-agent system operates inside a

virtual machine on the workstation and is runs in a continuous fashion instead of discrete time steps, and therefore a simulated hour requires an hour of actual runtime. This is exacerbated by the additional time required to extract the information from the agent output files and process the data. In order to improve this process a method of automating a test regime would prove useful, to implement such a system a supervisory program would need to be developed which could read from a list of test configurations and build the relevant agent population and models. The program would then insert the correct files into the main simulation directory to perform the desired test case, as illustrated in the following figure presented in Fig. 9.2.

The presented automation process requires a supervisory application which has the capability building files relevant to the set of cases needed and triggering the main simulation. The first step is to load the series of test configurations which define the number of tests to be implemented and the properties of each of those tests which may include the following:

Source Network – This defines which network model is the test to be performed on the supervisor program will set all references to components and model files within the gateway agent such that at runtime the correct.

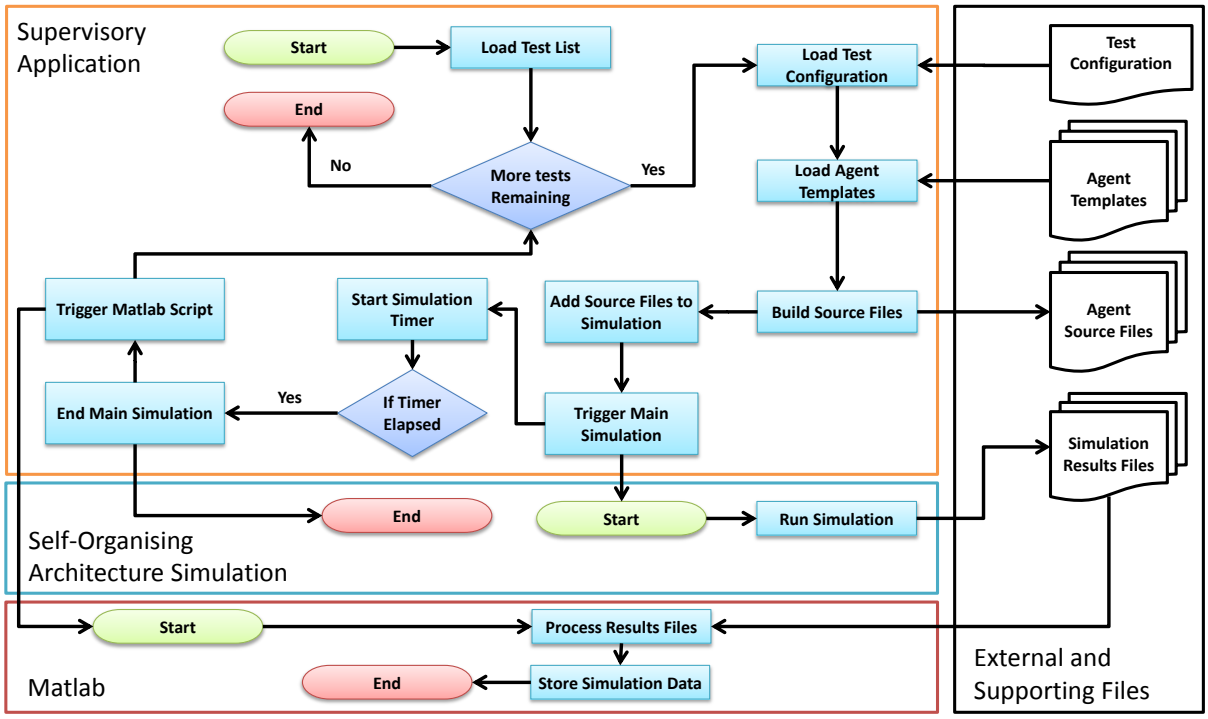


Fig. 9.2 – Structure of a System Automation Process

- **Agent Population –** This defines which agents are required for the simulation and the distribution of customers, aggregates and generator agents as contained within the components file. The supervisor would read the components file and determine which agents are needed.

- **Attack Conditions –** These are the settings applied by the error generator agent, determining which attack is to be applied and the associated severity of that attack, defining the number of attackers, targets, duration and quantity of attack events taking place within the simulation.

- **Network Properties -** Additional settings determining the number of customer agents willing to provide control and the level of control offered by those customers. These properties may also include variations on the control approach, thresholds in performance monitoring or in the decision making engine.

- **Architecture Properties –** The architecture properties include settings which define the starting architecture in the form of control placement, aggregate capacity and number of tiers. Settings may also involve alternate approaches to the decision making engine and transition approaches

- **Simulation Properties –** Finally the simulation settings contains parameters which sculpt the nature of the simulation, including the desired duration of the simulation, directories for output files and post-simulation data processing scripts.

Once the configuration for a given test scenario has been processed by the supervisor it will then need to construct an agent population which represents these settings. It will use the components file as a guide to the quantities and initial set points for the entities involved in the simulation. To perform this task the supervisor needs a set of agent templates, which it then uses to build the agent population by inserting the correct parameters loaded from the configuration into placeholders in the template. Completed agents are then placed inside the simulation directory before the supervisor triggers a simulation. Once the required runtime has elapsed the supervisor will then need to terminate the simulation and extract the results files generated by each of those agents such that they are not over-written by subsequent simulations. Finally the supervisor will need to call a Matlab script, to process the raw data files and produce a set of results.

9.6 FURTHER RESEARCH

While the self-organising architecture completed in the process of this research demonstrated the effectiveness of the concept and delivered improved resilience in the presence of an attack event, there are additional uses and scope for such a system. The following avenues for additional research building on the findings present cases for widening the scope of potential applications and lead to further developments.

9.6.1 Attack Diversity

The first research path considerers the possibility for increasing the attack diversity, in the current research the attack formats are based on low-rate denial of service attacks and operate through targeting a volume of attack traffic at crucial elements of the agent hierarchy. This strategy is a definitive attack on the communication network and leaves a trace in the form of data traffic anomalies in the form of increased congestion, unresponsiveness and reduced reactivity. Therefore the performance monitor has a signature to detect. Other attack formats will leave a different impact on the system and therefore require a different approach or an extension of the current decision making system. Therefore the self-organising architecture will need to be enhanced to include additional self-healing properties, these properties could include command validation stages, trust levels between sensors and data collection points and enhanced communication protocols.

Further attack formats may be launched against the electrical component of the cyber-physical system in the form or malware installed directly at the control point or the controller. In this case the network traffic may not exhibit any anomalies and therefore an additional level of continuous monitoring would need to be implemented on the various modelled electrical components to detect deviations. This would add physical network data to the performance monitoring input to the decision making engine and therefore more informed transitional events can be executed.

9.6.2 Control Diversity

The network under investigation in this research focussed on a voltage control problem and the control procedure is implemented through widespread demand side response. This solution was created to demonstrate the impact of attack formats on a command structure which relied on communication signals to trigger control actions and maintain demand reduction until such time it could be relieved. Additional control algorithms should be included in future developments of this work, so that the self-organising architecture can

be assessed when attempting to maintain multiple network variables. These control approaches may require the inclusion of additional controllable components within the network configurations including transformers, energy storage, soft open points and electric vehicles. A more diverse control base will increase the complexity and importance of the control signals flowing through the communication network. It will also increase the baseline level of traffic and lead to a system which is increasingly heavily loaded in the absence of an attack, therefore increasing the reliance on the self-organising architecture to reduce the computational burden. Furthermore if communication signals are lost or ignored as a result of attack traffic the consequences for these components could be more significant as these components may suffer damage if not controlled correctly.

In addition to introducing further algorithms to the simulation, additional control problems could be considered with differing consequences should the network under attack fail to mitigate the impacts of the attack. Scenarios can be evaluated such that the attack would lead to equipment damage or outages, and therefore presenting the case that a self-organising architecture could have economic value.

9.6.3 Network Diversity

A further approach would be to widen the range of networks upon which this concept is applicable – in the presented research the self-organising architecture is applied to the distribution level whereby smart-meters are determined to be the lowest level component. This is presented within the context of a radial distribution network composed of 340 customers across four feeders – even within this voltage level there is a large range of components, configurations and populations that can be considered. These additional configurations may be used for differing control scenarios or to evaluate differing attack strategies based on the components involved in the network.

A more rural network may contain fewer customers but would cover a larger geographical area with greater risk of interference from natural barriers. While these networks may be less likely to come under threat from a cyber-attack, the exposed nature of the components may present different challenges through failure and maintenance accessibility. In terms both of the electrical and communication networks. The performance monitoring aspect of the self-organising system may be more useful in feeding information into a maintenance schedule such that locations with limited accessibility which can only be observed remotely. Furthermore it may be the case that the architecture is applied to a higher voltage level

whereby instead of smart-meters being the lowest level component within the hierarchy, that role could be assumed by substations and the scope of the architecture could be translated to the transmission level. The core principles of the self-organising architecture presented in this work are applicable to differing levels of the power system. Agent based control mechanisms have been applied to microgrids and island systems and therefore adding a level of self-organisation would also be a feasible extension.

9.6.4 Co-Simulation

A final set avenue for further research would be to add to the simulation environment through the addition of modelling approaches for the communication layer. Interfacing a communication simulation tool with the agent architecture would enable the ability to model the latencies between nodes, and the properties of the ICT infrastructure, information which in turn could be used to inform the decision making aspect of the system through selecting agents closer to the point of failure in terms of communication distance to be promoted rather than estimating that distance through response times.

This approach would also allow experimentation through evaluating communication technologies and the effective range over which the system could be implemented – while the current test configuration considers a customer population 340 customers, the individual are less than a kilometre in length. Other environments may span a much wider area and being able to model the impacts of communicating with sensor and controllable entities which may have poor connectivity due to be being rural or under strong intermittent interference would add depth and applicability to the functions of the self-organising architecture.

9.6.5 Practical Experimentation

Aside from the set of expansions to the simulation element of the configuration, a different path would be to introduce hardware into the testing environment – which could be applied across differing aspects of the overall system. First of all considering the agents themselves and separating them from a single host configuration, multiple agents could be installed on raspberry pi style platforms. The interaction between physical components and simulated representations may deliver differing phenomena as a result of the messages being transmitted over a physical network and introducing additional properties involving actual bandwidth and communication delays.

A second avenue for introducing practical elements into the system is to include lab based testing in the form of physical or emulated controllable load, generation and storage devices. Therefore the operations completed by the architect agent aiming to perform actions to minimise commutative load and computational burden can be seen to impact on real world equipment.

9.7 SUMMARY

This chapter documented an evaluation of the processes involved with developing the self-organising architecture, discussing the design choices and potential limitations involved with the both the system itself and the test framework. Furthermore this chapter considered the potential implications for the research on the wider power systems community. Additionally the challenges involved with deploying such as system in practice were discussed from the perspective of the hardware, communication infrastructure and security concerns. Finally the chapter presented as series of potential development avenues if the self-organising architecture was to be enhanced for the purposes of improving future research applications and improving the evaluation and testing framework to facilitate additional experimentation.

Finally further research directions were suggested; directions which could further demonstrate and enhance the value of self-organising architectures for use in smart grid and over-arching power systems domains. Overall results were positive and exhibit encouraging properties which could be further developed through the means discussed in the chapter or used to build similar systems for alternate energy vectors or control problems.

Chapter 10: Conclusions

10.1 OVERVIEW

The research presented in this thesis began with an investigation of differing multi-agent architectures for the purposes of performing voltage control in a radial distribution network. The investigation concluded that no single static architecture proved to be the most resilient against all of the attack formats and therefore and indicated that a self-organising architecture would be an appropriate solution to improving resilience. A subsequent literature review supported the experimental evidence that self-organising architectures are more applicable when being used to improve robustness. Consequently a self-organising architecture was developed which featured three stages of operation. An initialisation state satisfied the self-organising requirement of establishing an endogenous global order, utilising mechanisms adapted from the EDETA and Tic-Tac-Toe Architecture solutions. Furthermore a performance monitoring stage allowed the architecture to be continuously aware of its state. Finally a fuzzy based decision making engine was implemented for the purposes of translating performance data into effective architecture transitions. When tested against a series of denial of service attacks, the self-organising architecture was able to reduce the impact of the attack on the communication variables and also prevented voltage control deterioration even under the most severe of tested attack formats.

This chapter outlines the key findings of the conducted research across the different stages of the investigation process; it also discusses how the research objectives presented in the first chapter have been fulfilled.

10.2 KEY FINDINGS

The key findings and the contributions of the investigations conducted throughout the course of this research are as follows.

1. The structure of a multi-agent architecture can influence its control and communication performance. Levels of congestion, reactivity, message efficiency and the ability to deliver control signals were all affected by the number of agents in the architecture and the distribution of aggregation agents. Across the 16 different control and communication architectures examined, no single design proved to be the most effective for all performance metrics. This was also reflected in response to handling an attack event, demonstrating that the robustness of a system was also linked to the architecture design. Therefore a self-organising architecture which

could switch between configurations would be beneficial, especially with respect to resistance to attacks.

2. A denial of service attack is able to prevent the dissemination of control signals as a result of increasing message congestion at the controller layer by up to 16,000 messages. Consequently response times between controller and customer increased by up to 8.2 seconds, therefore severing a control connection and triggering control deterioration.

3. A novel self-organising architecture was developed for performing voltage control in smart grids, adapting techniques from wireless communication, vehicle and sensor networks. The developed architecture featured three operational stages and could construct a stable starting configuration.

4. A fuzzy based decision making engine was developed which could analyse several different performance metrics normalised into a single computational burden indicator. The decision making engine could translate the indicator and its rate of change into an architectural transition action.

5. Monitoring and responding to issues within the communication layer of a cyber-physical smart grid can result in beneficial performance outcomes for control of physical components.

6. The use of self-organisation is appropriate and beneficial for use within an agent based network management system with respect to initialising a collection of agents, forming connections between architectural tiers and preventing control deterioration in the presence of a denial of service attack containing up to 24% of the customer population.

7. The developed self-organising architecture was capable of reducing the computational burden indicator by at least 64% and by over 90% in the majority of examples. Therefore indicating applicability beyond cyber-security in processing heavily loaded smart grid communication architectures.

10.3 FULLFILLMENT OF RESEARCH OBJECTIVES

Three research objectives were set at the beginning of this thesis where are as follows:

1. Evaluate comparative performances across differing control and communication architectures in the context of distribution network management with a view to determining the potential role for implementing self-organisation. This investigation aimed to determine what the benefits would be of providing self-organisation within the control and communication architecture and why self-organisation is an appropriate approach for cyber-security.

The first of the research objectives considered the evaluation of a range of control and communication architectures involved with performing voltage control. The aim of the evaluation was to determine the performance differences between the differing architectures with respect to robustness to an attack event. Additional communication level performance was also considered in terms of message congestion, reactivity, message efficiency – and the ability to perform the voltage control objective. This investigation discovered that, the structure of an architecture has a strong influence on the control, communication and robustness performance, structures with a greater concentration of data collection points exhibited stronger performance in terms of message congestion and reactivity. Architectures with fewer aggregation agents performed stronger in terms of message efficiency. In addition to the structure of the architecture, increasing the agent population also created variations in performance in each of the examined criteria, as scalability improved through increasing the number of aggregates.

Overall the results demonstrated that no single architecture design can deliver the strongest performance across all scales and all of the recorded metrics. A total of 16 alternate control and communication configurations were implemented and examined against communication performance, control performance and robustness performance. Out of the total set of configurations only the disaggregated architecture demonstrated poor performance across a wider range of test scenarios and therefore was not considered for use in the self-organising system development. This illustrates that the presence of data aggregation points is crucial in developing smart grid agent architectures. The research discovered that the most prominent driver for implementing self-organisation was in response to failure or in the event of a cyber-attack, which was supported by self-organising applications presented in literature as documented in chapter 4.

The conclusion for the first research objective was that there is definitive value in the use of self-organising architectures specifically in response to unexpected events affecting the agent population. A planning stage could design an architecture to favour specific properties relevant to the control objective of the agent population or to account for an estimated volume of communication load. However a cyber-attack event cannot be adequately planned for and therefore developing an architecture with the capability to respond to such an event would be an advantage.

2. Develop and implement an agent population with functioning self-organisational properties including architecture formation, contemporaneous monitoring and decision making.

The second objective required the development of a self-organising architecture with the ability to determine its own initial configuration based on a set of input parameters, perform continuous performance monitoring and effect appropriate reconfiguration actions. The comparative metrics implemented in the fulfilment of objective 1, were translated into performance monitoring criteria which acted as inputs to a decision making engine. The development process was successful as all of the desired functions were implemented and demonstrated to be functional. Furthermore the fulfilment of the objective illustrated that developing an agent based self-organising architecture was feasible with current technologies, and therefore future implementations have the potential for further performance gains.

Three stages of operation were developed as part of the self-organising architecture, the first of which was an initialisation stage which served as a functional method of connecting customer agents with controllers and being able to distribute the connections between individual controllers. This process was able to assign each customer and generation agent a valid connection and no agents were left isolated during initialisation. Secondly a performance monitoring stage extracted relevant information from the agent population relating to communication performance. Individual agents monitored their own parameters and any observed anomalies were reported to the architect agent, this system allowed efficient delivery of error reports and a flexible approach to incorporating future performance monitoring criteria. A final stage involved a fuzzy based decision making engine which translated performance data into transitions which could be successfully

triggered by the architect agent. Each transition event formed a stable configuration and no connections were dropped during the process and no communication signals were disrupted.

Overall this objective was fulfilled, as the implemented self-organising architecture was able to perform the three stages of operation successfully, furthermore the agent population interfaced with an external load flow engine modelling an electrical network. This therefore ensured that the self-organising architecture was also capable of performing voltage control and customer data collection in addition the self-organising properties.

3. Examine the performance of the developed self-organised system in the presence of external network threats in the form of cyber-attack events with respect to control and communication performance. These performances are also examined with respect to a static architecture undergoing the same cyber-attack conditions with the objective of learning which variables are affected by an ongoing attack. A further learning outcome is to identify whether a communication variables have an impact on the electrical performance of a network while under attack. To determine whether the self-organising architecture can improve electrical performance by improving communication layer performance.

The final objective evaluated the self-organising architecture when it was exposed to a series of low rate denial of service attack events. From the series of tests applied to the self-organising architecture it was concluded that smaller attack events, whereby 6% of the customer population were perpetrating the denial of service attack the attack was not significant enough to warrant the use of architecture transitions. As the attack was scaled up to involve 24% of the customer population, without the assistance of self-organisation control was lost to 50% of the electrical network under investigation. Whereas with the use of self-organisation all customers received control signals to correct the voltage deviation, demonstrating that the self-organising approach was capable of maintaining controllability during a denial of service attack event. Additionally the results demonstrated that the self-organising architecture was able to reduce the computational burden by at least 64% and by 90% in the majority of cases, therefore illustrating that the performance advantages of the self-organising architecture extend beyond the electrical layer. Finally these figures indicated the correlation between correcting issues within the communication layer and the resulting positive impacts on control performance.

10.4 SUMMARY

In summary this research has demonstrated that the vision of increasing the amount observability and controllability within future smart grids and enhancing the cyber-physical infrastructure brings with it a set of new challenges. These challenges arise from the number of components communicating data and requiring control signals, this in turn increases amount of data flowing through the communication infrastructure. With the increased reliance on ICT components, there is the further challenge created through the risk of cyber-threats against the power system. The research determined that the application of a self-organising architecture is an effective mechanism for reducing the consequences of facing these challenges and one which offers the flexibility associated with future network concepts. The work illustrated that a static architecture could suffer control deterioration as a result of a denial of service attack, whereas a self-organising architecture could continue to achieve the control objective. The self-organising approach has considerable room for further research, especially with respect to processing additional cyber-threats which remain an interesting and expanding research topic and where self-organising architectures could one of the defensive tools in future developments.

References

[1] S. McArthur, P. Taylor, G. Ault, J. King, D. Athanasiadis, V. Alimisis and M. Czaplewski, "The Autonomic Power System - Network operation and control beyond smart grids," in *3rd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies (ISGT Europe), 2012* , Berlin, 2012.

[2] W. v. d. Hoek and M. Wooldridge, "Multi-Agent Systems," in *Handbook of Knowledge Representation*, Elselvier , 2008, pp. 887-928.

[3] G. Ali, N. A. Shaikh and A. W. Shaikh, "A Research Survey of Software Agents and Implementation Issues in Vulnerability Assessment and Social Profiling Models," *Australian Journal of Basic and Applied Sciences,* vol. 4, no. 3, pp. 442-449, 2010.

[4] N. R. Jennings and M. Wooldridge, "Applications of Intelligent Agents," http://maya.cs.depaul.edu/~classes/cs480/readings/applications-of-intelligent-agents-jennings-wooldridge.pdf, London, 1998.

[5] Q. H. Mahmoud, "Software Agents: Characteristics and Classification," http://www.cis.uoguelph.ca/~qmahmoud/post/agentsoft.pdf.

[6] G. M. Marakas, "Chapter 17: Intelligent Software Agents," in *Decision Support Systems in the 21st Century*, Second ed., Prentice-Hall, 2003.

[7] R. Deters, "Scalability & Multi-Agent Systems," in *2 nd International Workshop Infrastructure for Agents, MAS and Scalable MAS. 5th Int. conference on Autonomous Agents*, 2001.

[8] A.-R. A. Khatib, *Internet--based Wide Area Measurement Applications in Deregulated Power Systems,* Blacksburg, Virginia: PhD Thesis: Virginia Polytechnic and State University, 2002.

[9] J. Solanki and N. Schulz, "Multi-Agent System for Islanded Operation of Distribution Systems," Power Systems Conference and Exposition, 2006. PSCE '06. 2006 IEEE PES. , 2006.

[10] H. Al-Mohannadi, Q. Mirza and A. Namanya, "Cyber-Attack Modeling Analysis Techniques: An Overview," in *IEEE International Conference on Future Internet of Things and Cloud Workshops (FiCloudW),* , Vienna, 2016.

[11] G. Cisotto and L. Badia, "Cyber security of smart grids modeled through epidemic models in cellular automata," in *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM),* , Coimbra, Portugal, 2016.

[12] G. N. Ericsson, "Cyber Security and Power System Communication—Essential Parts of a Smart Grid Infrastructure," *IEEE Transactions on Power Delivery,* vol. 25, no. 3, pp. 1501-1507, 2010.

[13] F. Cohen, "The Smarter Grid," *IEEE Security & Privacy,* vol. 8, no. 1, pp. 60-63, 2010.

[14] S. Li, Y. Yılmaz and X. Wang, “Quickest Detection of False Data Injection Attack in Wide-Area Smart Grids,” *IEEE Transactions on Smart Grid,* vol. 6, no. 6, pp. 2725 - 2735, 2015.

[15] L. Jia, J. Kim, R. J. Thomas and L. Tong, “Impact of Data Quality on Real-Time Locational Marginal Price,” *IEEE Transactions on Power Systems,* vol. 29, no. 2, pp. 627 - 636, 2014.

[16] X. Liu, Z. Bao, D. Lu and Z. Li, “Modeling of Local False Data Injection Attacks With Reduced Network Information,” *IEEE Transactions on Smart Grid,* vol. 6, no. 4, pp. 1686 - 1696, 2015.

[17] J. Hong, C.-C. Liu and M. Govindarasu, “Integrated Anomaly Detection for Cyber Security of the Substations,” *IEEE Transactions on Smart Grid,* vol. 5, no. 4, pp. 1643 - 1653, 2014.

[18] Y. Yang, K. McLaughlin, S. Sezer, T. Littler, E. G. Im, B. Pranggono and H. F. Wang, “Multiattribute SCADA-Specific Intrusion Detection System for Power Networks,” *IEEE Transactions on Power Delivery,* vol. 29, no. 3, pp. 1092-1102, 2014.

[19] Y. Zhang, L. Wang, Y. Xiang and C.-W. Ten, “Power System Reliability Evaluation With SCADA Cybersecurity Considerations,” *IEEE Transactions on Smart Grid,* vol. 6, no. 4, pp. 1707 - 1721, 2015.

[20] J. Bialek, “Recent failures - Recent failures in US/Canada, Scandinavia and Italy,” in *How Secure are Britain's Electricity Supplies?*, London, 2004.

[21] Y. Chakhchoukh and H. Ishii, “Coordinated Cyber-Attacks on the Measurement Function in Hybrid State Estimation,” *IEEE Transactions on Power Systems,* vol. 30, no. 5, pp. 2487 - 2497, 2015.

[22] S. Mousavian, J. Valenzuela and J. Wang, “A Probabilistic Risk Mitigation Model for Cyber-Attacks to PMU Networks,” *IEEE Transactions on Power Systems,* vol. 30, no. 1, pp. 156 - 165, 2014.

[23] H. M. Khalid and J. C. H. Peng, “A Bayesian Algorithm to Enhance the Resilience of WAMS Applications Against Cyber Attacks,” *IEEE Transactions on Smart Grid,* vol. 7, no. 4, pp. 2026-2037, 2016.

[24] Y. Isozaki, S. Yoshizawa, Y. Fujimoto, H. Ishii, I. Ono, T. Onoda and Y. Hayashi, “Detection of Cyber Attacks Against Voltage Control in Distribution Power Grids With PVs,” *IEEE Transactions on Smart Grid,* vol. 7, no. 4, pp. 1824 - 1835, 2016.

[25] K. R. Davis, C. M. Davis, S. A. Zonouz, R. B. Bobba, R. Berthier, L. Garcia and P. W. Sauer, “A Cyber-Physical Modeling and Assessment Framework for Power Grid Infrastructures,” *IEEE Transactions on Smart Grid,* vol. 6, no. 5, pp. 2464 - 2475, 2015.

[26] R. Liu, C. Vellaithurai, S. S. Biswas, T. T. Gamage and A. K. Srivastava, “Analyzing the Cyber-Physical Impact of Cyber Events on the Power Grid,” *IEEE Transactions on Smart Grid,* vol. 6, no. 5, pp. 2444 - 2453, 2015.

[27] C. Vellaithurai, A. Srivastava, S. Zonouz and R. Berthier, “CPIndex: Cyber-Physical Vulnerability Assessment for Power-Grid Infrastructures,” *IEEE Transactions on Smart Grid,* vol. 6, no. 2, pp. 566-575, 2015.

[28] D. Kushner, “The Real Story of Stuxnet,” *IEEE Spectrum,* pp. Online: http://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet, 2013 Feburary 26.

[29] D. Storm, “Computer World,” 27 January 2016. [Online]. Available: http://www.computerworld.com/article/3026609/security/no-israels-power-grid-wasnt-hacked-but-ransomware-hit-israels-electric-authority.html. [Accessed 30 July 2016].

[30] Electricity Information Sharing and Analysis Centre, “Analysis of the Cyber Attack on the Ukranian Power Grid,” E-ISAC. Available online: https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf, 2016.

[31] L. Jun, Z. Shunyi, Z. Zailong and W. Pan, “A Novel Network Management Architecture for Self-organizing Network,” in *International Conference on Networking, Architecture, and Storage, 2007. NAS 2007.* , Guilin, 2007.

[32] J. Magee and J. Kramer, “Dynamic structure in software architectures,” in *In Proceedings of the Fourth ACM SIGSOFT Symposium on the Foundations of Software Engineering*, 1996.

[33] J. Magee and J. Kramer, “Self organising software architectures,” in *ISAW '96 Joint proceedings of the second international software architecture workshop (ISAW-2)*, 1996.

[34] W. Lu, Y. Gu, R. Prasad, A. Lo and I. Niemegeers, “A Self-organized Personal Network Architecture,” in *Third International Conference on Networking and Services, 2007. ICNS.*, 2007.

[35] M. Niazi and S. Laghari, “An Intelligent Self-Organizing Power-Saving Architecture: An Agent-based Approach,” in *2012 Fourth International Conference on Computational Intelligence, Modelling and Simulation (CIMSiM)*, Kuantan, 2012.

[36] M. Cherif, S. Senouci and B. Ducourthial, “Vehicular network self-organizing architectures,” in *2009 5th IEEE GCC Conference & Exhibition* , Kuwait City, 2009.

[37] O. Aliu, A. Imran, M. Imran and B. Evans, “A Survey of Self Organisation in Future Cellular Networks,” *IEEE Communications Surveys & Tutorials,* vol. 15, no. 1, pp. 336 - 361, 2013.

[38] G. D. M. Serugendo, M.-P. Gleizes and A. Karageorgos, “Self-organization in multi-agent systems,” *Knowl. Eng. Rev.,* vol. 20, no. 2, pp. 165-189, 2005.

[39] A. Abdrabou, “A Wireless Communication Architecture for Smart Grid Distribution Networks,” *IEEE Systems Journal,* vol. PP, no. 99, pp. 1 - 11, 2014.

[40] S. Bavarian, L. Lampe, C. Siew, S. Lancashire and K. Adeleye, “Leveraging the smart metering infrastructure in distribution automation,” in *2012 IEEE Third International Conference on Smart Grid Communications*, Tainan, 2012.

[41] S. Seidi Khorramabadi and A. Bakhshai, “Intelligent Control of Grid-connected Microgrids: An Adaptive Critic-based Approach,” *IEEE Journal of Emerging and Selected Topics in Power Electronics,* vol. 99, no. PP, p. 1, 2014.

[42] A. Godarzi, S. Niaki, F. Ahmadkhanlou and R. Iravani, “Local and global optimization of exportable vehicle power based smart microgrid,” in *2011 IEEE PES Innovative Smart Grid Technologies (ISGT),* , Anaheim, CA, 2011.

[43] P. Vrba, V. Marik, P. Siano, P. Leitao, G. Zhabelova, V. Vyatkin and T. Strasser, “A Review of Agent and Service-Oriented Concepts Applied to Intelligent Energy Systems,” *IEEE Transactions on Industrial Informatics,* vol. 10, no. 3, pp. 1890 - 1903, 2014.

[44] T. Logenthiran, D. Srinivasan, A. Khambadkone and H. Aung, “Scalable Multi-Agent System (MAS) for operation of a microgrid in islanded mode,” in *2010 Joint International Conference on Power Electronics, Drives and Energy Systems (PEDES) & 2010 Power India,* , New Delhi, 2010.

[45] L. Korba and R. Song, “Modelling and Simulating the Scalability of a Multi-Agent Application System,” National Reseach Council of Canada, 2002.

[46] K. Chmiel, M. Gawinecki, P. Kaczmarek, M. Szymczak, M. Paprzycki, B. C. Pierce and D. N. Turner, “Efficiency of the JADE agent platform,” *Scientific Programming,* vol. 12, no. 2, pp. 159-172 , 2005.

[47] L. C. Lee, H. S. Nwana, D. T. Ndumu and P. D. Wilde, “The stability, scalability and performance of multi-agent Systems,” *BT Technology J,* vol. 16, no. 3, pp. 94-103, 1998.

[48] G. McKinstry, S. Galloway and I. Kockar, “An initial assessment of the potential impact of smart metering on a decentralised energy network,” in *2010 45th International Universities Power Engineering Conference*, Cardiff, Wales, 2010.

[49] V. Alimisis, C. Piacentini, J. King and P. Taylor, “Operation and Control Zones for Future Complex Power Systems,” in *2013 IEEE Green Technologies Conference*, Denver, CO, 2013.

[50] P. Papadopoulos, N. Jenkins, L. Cipcigan, I. Grau and E. Zabala, “Coordination of the Charging of Electric Vehicles Using a Multi-Agent System,” *IEEE Transactions on Smart Grid,* vol. 4, no. 4, pp. 1802-1809, 2013.

[51] C.-X. Dou and B. Liu, “Multi-Agent Based Hierarchical Hybrid Control for Smart Microgrid,” *IEEE Transactions on Smart Grid,* vol. 4, no. 2, pp. 771-778, 2013.

[52] S. D. J. McArthur, E. M. Davidson, V. M. Catterson, A. L. Dimeas, N. D. Hatziargyriou, F. Ponci and T. Funabashi, "Multi-Agent Systems for Power Engineering Applications—Part I: Concepts, Approaches, and Technical Challenges," *Transactions on Power Systems,* vol. 22, no. 4, pp. 1743-1752, 2007.

[53] S. McArthur, E. Davidson, V. Catterson, A. Dimeas, N. Hatziargyriou, F. Ponci and T. Funabashi, "Multi-Agent Systems for Power Engineering Applications—Part II: Technologies, Standards, and Tools for Building Multi-agent Systems," *IEEE Transactions On Power Systems,* vol. 22, no. 4, pp. 1753 - 1759, 2007.

[54] T. Logenthiran, N. U. o. S. T. Dept. of Electr. & Comput. Eng., D. Srinivasan, A. Khambadkone and H. Aung, "Scalable Multi-Agent System (MAS) for operation of a microgrid in islanded mode," in *2010 Joint International Conference on Power Electronics, Drives and Energy Systems (PEDES) & 2010 Power India*, New Delhi, 2010.

[55] E. Cortese, F. Quarta and G. Vitaglione, "Scalability and performance of jade message transport system," in *Autonomous Agents and Multiagent Systems (AAMAS)*, Bologna, 2002.

[56] A. Prostejovsky, W. Lepuschitz, T. Strasser and M. Merdan, "Autonomous service-restoration in smart distribution grids using Multi-Agent Systems," 2012 25th IEEE Canadian Conference on Electrical & Computer Engineering (CCECE), 2012.

[57] M. Manbachi, M. Nasri, B. Shahabi, H. Farhangi, A. Palizban, S. Arzanpour, M. Moallem and D. Lee, "Real-Time Adaptive VVO/CVR Topology Using Multi-Agent System and IEC 61850-Based Communication Protocol," *IEEE Transactions on Sustainable Energy,* vol. 5, no. 2, pp. 587-597, 2014.

[58] C. Colson and M. Nehrir, "Comprehensive Real-Time Microgrid Power Management and Control With Distributed Agents," *IEEE Transactions on Smart Grid,* vol. 4, no. 1, pp. 617-627, 2013.

[59] P. Nguyen, W. Kling and P. Ribeiro, "A Game Theory Strategy to Integrate Distributed Agent-Based Functions in Smart Grids," *IEEE Transactions on Smart Grid,* vol. 4, no. 1, pp. 568-576, 2013.

[60] F. Ren, M. Zhang and D. Sutanto, "A Multi-Agent Solution to Distribution System Management by Considering Distributed Generators," *IEEE Transactions on Power Systems,* vol. 28, no. 2, pp. 1442-1451, 2013.

[61] E. Karfopoulos and N. Hatziargyriou, "A Multi-Agent System for Controlled Charging of a Large Population of Electric Vehicles," *IEEE Transactions on Power Systems,* vol. 22, no. 2, pp. 1196-1204, 2013.

[62] Y. Xu, W. Liu and J. Gong, "Stable Multi-Agent-Based Load Shedding Algorithm for Power Systems," *IEEE Transactions on Power Systems,* vol. 26, no. 4, pp. 2006-2014, 2011.

[63] B. Horling and V. Lesser, “A Survey of Multi-Agent Organizational Paradigms,” *The Knowledge Engineering Review,* pp. 281-316, 2005.

[64] E. Negeri, N. Baken and M. Popov, “Holonic Architecture of the Smart Grid,” *Smart Grid and Renewable Energy,* vol. 4, no. 2, pp. 202-212, 2013.

[65] J. Zhou, R. Hu and Y. Qian, “Scalable Distributed Communication Architectures to Support Advanced Metering Infrastructure in Smart Grid,” *IEEE Transactions on Parallel and Distributed Systems,* vol. 23, no. 9, pp. 1632-1642, 2012.

[66] Y. Wang, P. Yemula and A. Bose, “Decentralized Communication and Control Systems for Power System Operation,” *IEEE Transactions on Smart Grid,* vol. 6, no. 2, pp. 885-893, 2014.

[67] J. Pérez, J. Díaz, C. Vidal, D. Rodríguez and D. Fernández, “Self-Balancing Distributed Energy in Power Grids: An Architecture Based on Autonomic Computing,” in *2014 47th Hawaii International Conference on System Sciences (HICSS)* , Waikoloa, HI, 2014.

[68] T. Nagata, Y. Tao, H. Sasaki and H. Fujita, “A multiagent approach to distribution system restoration,” in *IEEE Power Engineering Society General Meeting, 2003,*, 2003.

[69] V. Giordano, F. Gangale, G. Fulli and M. S. Jiménez, “Smart Grid projects in Europe: lessons learned and current developments,” European Commission, Joint Research Centre: Institute for Energy, Available online: https://ses.jrc.ec.europa.eu/sites/ses/files/documents/smart_grid_projects_in_europe.pdf, 2011.

[70] Scottish and Southern Energy, *NINES: Northern Isles New Energy Solutions - Overview Presentation,* Available Online: http://www.ninessmartgrid.co.uk/wp-content/uploads/2013/05/NINES-Project-Overview-Presentation.pdf, 2012.

[71] G. Ault, D. Frame, S. Gill, I. Kockar, M. Dolan, O. Anaya-Lara, S. Galloway, B. O'Neill, C. Foote and A. Svalovs, “Northern Isles New Energy Solutions: Active network management stability limits,” in *2012 3rd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies (ISGT Europe)*, Berlin, 2012.

[72] K. Svehla, “Home space and water heatiner aspects of the SSE Shetland NINES Project,” University of Strathclyde Energy Systems Research Unit: Available Online: http://www.ninessmartgrid.co.uk/wp-content/uploads/2013/06/Home-Space-and-Water-Heating-Aspects-of-the-SSE-Shetland-NINES-Project.pdf, 2012.

[73] Electricity North West, “Low Votlage Network Solutions: A First Tier Low Carbon Networks Fund Project. Closedown Report,” Electicity North West. Available Online: http://www.enwl.co.uk/docs/default-source/future-low-voltage/lv-network-solutions-close-down-june-14-v4-1-final.pdf?sfvrsn=2, 2014.

[74] T. Gozel and A. Navarro, “Deliverable 3.5 “Creation of aggregated profiles with and without new loads and DER basedon monitored data”,” University of Manchester: Available Online - http://www.enwl.co.uk/docs/default-source/future-low-voltage/university-of-manchester-appendix-h.pdf?sfvrsn=2, 2014.

[75] S. Kaushik, P. Bale, R. Aggarwal, M. Dale, M. Redfern and A. Smyth, “Project SoLa BRISTOL and the “ecohome”,” in *2013 48th International Universities' Power Engineering Conference (UPEC)*, Dublin, 2013.

[76] S. Kaushik, M. Dale, R. Aggarwal, A. Smyth, M. Redfern and K. Waite, “Project SoLa BRISTOL migration from "ecohome" to "integrated homes",” in *2014 49th International Universities Power Engineering Conference (UPEC)*, Cluj-Napoca, 2014.

[77] Low Carbon Networks Fund, “Low Carbon Networks Fund Full Submission Pro Forma,” Low Carbon Networks Fund - Available Online: https://www.ofgem.gov.uk/ofgem-publications/91889/appendix7publish.pdf, 2011.

[78] GRID4EU coordination team, “Grid4EU Innovation for Energy Networks,” GRID4EU coordination team, Available Online: http://grid4eu.blob.core.windows.net/media-prod/29375/grid4eu-final-report_normal-res.pdf, 2016.

[79] Grid4EU , “dD1.1 Demo 1 - Advanced MV network operations using a multi-agent system,” Grid4EU, Available Online: http://grid4eu.blob.core.windows.net/media-prod/6578/Grid4EU_dD1.1_Demo_1_V1.0.pdf, 2012.

[80] Web2Energy, “Managemt summary of the project Web2Energy,” 2012. [Online]. Available: https://www.web2energy.com/results/management-summary/?L=%2Fproc%2Fself%2Fenviron%3Fprint%3D1%3Fprint%3D1%3Fprint%3D1%3Fprint%3D1. [Accessed 28 Feburary 2017].

[81] UPGrid Project Consortium, “Internet of Energy,” Internet of Energy, Available Online: http://upgrid.eu/wp-content/uploads/2016/04/UPGRID_Project-Presentation.pdf, 2016.

[82] P. M. Nunes, P. G. Matos, P. Pereira, P. Felício, A. Botelho, J. Guisado, G. Pires, J. Moreira and Y. Ahmad, “Upgrid Portuguese demo — Market challenges (in) the power grid,” in *Cired Workshop*, Helsinki, 2016.

[83] A. Blaver and P. Italiano, “Perth Solar City Annual Report,” Australian Government Smart Cities, Perth. Available Online: http://perthsolarcity.com.au/resources/Perth_Solar_City_Annual_Report_2012_low_res.pdf, 2012.

[84] Australian Goverment: Department of Resources Energy and Tourism, “Final Report of the Lessons and Highlights of the Solar Cities Program,” Australian Goverment: Department of

Resources Energy and Tourism, Available Online: https://industry.gov.au/Energy/EnergyEfficiency/Documents/solar-cities-journey.pdf, 2013.

[85] AEP Ohio, “Final Technical Report,” AEP Ohio, Available Online: https://www.smartgrid.gov/files/AEP_Ohio_DE-OE-0000193_Final_Technical_Report_06-23-2014.pdf, 2014.

[86] K. V. Meter, “Moving from Conventional to Real Time Cyber-Security,” Lockheed Martin, Available Online: http://www.aertc.org/conference2010/speakers/AEC%202010%20Session%202/2C%20Smart%20Networks%20II%20Cybersecurity/Kenneth%20Van%20Meter/van%20meter%20presSECURED.pdf, 2010.

[87] Symantec, “2016 Internet Security Threat Report - Volume 21 April 2016,” Symantec, 2016.

[88] Y. Liu, P. Ning and M. K. Reiter, “False data injection attacks against state estimation in electric power grids,” in *Proceedings of the 16th ACM conference on Computer and communications security*, New York, 2009.

[89] J. Kim and L. Tong, “On Topology Attack of a Smart Grid: Undetectable Attacks and Countermeasures,” *IEEE Journal on Selected Areas in Communications,* vol. 31, no. 7, pp. 1294 - 1305, 2013.

[90] X. Liu and Z. Li, “False Data Attacks Against AC State Estimation With Incomplete Network Information,” *IEEE Transactions on Smart Grid ,* vol. PP, no. 99, pp. 1-10, 2016.

[91] O. Kosut, L. Jia and R. J. Thomas, “Limiting false data attacks on power system state estimation,” in *44th Annual Conference on Information Sciences and Systems (CISS), 2010 44th Annual Conference on*, Princeton University, 2010.

[92] ICS-CERT, “ICS Focused Malware (Update A),” 27 June 2014. [Online]. Available: https://ics-cert.us-cert.gov/alerts/ICS-ALERT-14-176-02A. [Accessed 3 10 2016].

[93] ICS-CERT, “Ongoing Sophisticated Malware Campaign Compromising ICS (Update E),” 2 March 2016. [Online]. Available: https://ics-cert.us-cert.gov/alerts/ICS-ALERT-14-281-01B. [Accessed 3 October 2016].

[94] D. Allan, “Was attack on BBC website the biggest volley of DDoS fire ever seen?,” 11 January 2016. [Online]. Available: http://www.techradar.com/news/internet/attack-against-bbc-website-was-the-biggest-volley-of-ddos-fire-ever-seen--1312864. [Accessed 3 October 2016].

[95] A. Kuzmanovic and E. W. Knightly, “Low-rate TCP-targeted denial of service attacks and counter strategies,” *IEEE/ACM Transactions on Networking ,* vol. 14, no. 4, pp. 683 - 696, 2006.

[96] J. Luo, X. Yang, J. Wang, J. Xu, J. Sun and K. Long, “On a Mathematical Model for Low-Rate Shrew DDoS,” *IEEE Transactions on Information Forensics and Security,* vol. 9, no. 7, pp. 1069 - 1083, 2014.

[97] S. Ingram, S. Probert and K. Jackson, “The Impact of Small Scale Embedded Generation on the Operating Parameters of Distribution Networks,” PB Power, 2003.

[98] Customer-Led Network Revolution, “Customer-Led Network Revolution,” [Online]. Available: http://www.networkrevolution.co.uk/resources/project-data/.

[99] National Grid, *Grid Code Issue 5 Revision 7,* Available from: http://www2.nationalgrid.com/WorkArea/DownloadAsset.aspx?id=32449: National Grid, 2014.

[100] IEEE-SA Standards Board, “IEEE Recommended Practice for Monitoring Electric Power Quality,” IEEE, 2009.

[101] C. D. Cameron, P. C. Taylor and C. Patsios, “On The Benefits of Using Self-Organising Multi-Agent Architectures in Network Management,” in *2015 International Symposium on Smart Electric Distribution Systems and Technologies (EDST)*, Vienna, 2015.

[102] U. Richter, M. Mnif, J. Branke, C. Müller-Schloer and H. Schmeck, “Towards a generic observer/controller architecture for Organic Computing,” in *Informatik für Menschen - Band 1, GI-Edition*, B. K. Verlag, Ed., 2006, pp. 112-119.

[103] R. Pathak, P. Hu, J. Indulska, M. Portmann and W. L. Tan, “Towards efficient opportunistic communications: A hybrid approach,” in *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops),* , San Diego, CA, 2013.

[104] T. Harrold and A. Nix, “Capacity enhancement using intelligent relaying for future personal communication systems,” in *52nd Vehicular Technology Conference, 2000. IEEE-VTS Fall VTC 2000.* , Boston, MA, 2000.

[105] P. Kulkarni, S. Gormus, Z. Fan and B. Motz, “A self-organising mesh networking solution based on enhanced RPL for smart metering communications,” in *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM),*, Lucca, 2011.

[106] R. Litjens, F. Gunnarsson, B. Sayrac, K. Spaey, C. Willcock, A. Eisenblatter, B. Gonzalez Rodriguez and T. Kurner, “Self-Management for Unified Heterogeneous Radio Access Networks,” in *2013 IEEE 77th Vehicular Technology Conference (VTC Spring),*, Dresden, 2013.

[107] ETSI - European Telecommunications Standards Institute, “3GPP TS 32.101: Telecommunication management; Principles and high level requirements,” ETSI - European Telecommunications Standards Institute, Available Online: http://www.etsi.org/deliver/etsi_ts/132100_132199/132101/12.00.00_60/ts_132101v120000p.pdf, 2014.

[108] A. Eisenblatter, B. Gonzalez Rodriguez, F. Gunnarsson, T. Kurner, R. Litjens, B. Sas, B. Sayrac, L. Schmelz and C. Willcock, “Integrated self-management for future radio access networks: Vision and key challenges,” in *2013 Future Network and Mobile Summit (FutureNetworkSummit)*, Lisboa, 2013.

[109] C. Frenzel, S. Lohmuller and L. Schmelz, “Dynamic, context-specific SON management driven by operator objectives,” in *2014 IEEE Network Operations and Management Symposium (NOMS),* , Krakow, 2014.

[110] ETSI - European Telecommunications Standards Institute, “Telecommunication management; Self-Organizing Networks (SON) Policy Network Resource Model (NRM) Integration Reference Point (IRP); Information Service (IS). Version 11.7.0,” ESTI, Available online:http://www.etsi.org/deliver/etsi_ts/132500_132599/132522/11.07.00_60/, 2013.

[111] T. Ojha, M. Khatua and S. Misra, “Tic-Tac-Toe-Arch: a self-organising virtual architecture for Underwater Sensor Networks,” *Wireless Sensor Systems, IET,* vol. 3, no. 4, pp. 307-316, 2013.

[112] J. Wang, D. Li, M. Zhou and D. Ghosal, “Data Collection with Multiple Mobile Actors in Underwater Sensor Networks,” in *ICDCS '08. 28th International Conference on Distributed Computing Systems Workshops, 2008.*, Beijing, 2008.

[113] S. Roy, P. Arabshahi, D. Rouseff and W. Fox, “Wide area ocean networks: architecture and system design considerations,” in *WUWNet '06 Proceedings of the 1st ACM international workshop on Underwater networks*, Los Angeles, CA, 2006.

[114] J. Capella, A. Bonastre, R. Ors and S. Climent, “A New Energy-Efficient, Scalable and Robust Architecture for Wireless Sensor Networks,” in *2009 3rd International Conference on New Technologies, Mobility and Security (NTMS),* , Cairo, 2009.

[115] J. Capella, A. Bonastre, J. Serrano and R. Ors, “A New Robust, Energy-efficient and Scalable Wireless Sensor Networks Architecture Applied to a Wireless Fire Detection System,” in *WNIS '09. International Conference on Wireless Networks and Information Systems, 2009.*, Shanghai, 2009.

[116] S. Climent, J. V. Capella, N. Meratnia and J. J. Serrano, “Underwater Sensor Networks: A New Energy Efficient and Robust Architecture,” *Sensors ,* vol. 12, no. 1, pp. 704-731, 2012.

[117] F. Atero, J. Vinagre, J. Ramiro and M. Wilby, “A low energy and adaptive routing architecture for efficient field monitoring in heterogeneous wireless sensor networks,” in *2011 International Conference on High Performance Computing and Simulation (HPCS),*, Istanbul, 2011.

[118] W. Narzt, U. Wilflingseder, G. Pomberger, D. Kolb and H. Hortner, “Self-organising congestion evasion strategies using ant-based pheromones,” *IET Intelligent Transport Systems,* vol. 4, no. 1, pp. 93-102, 2010.

[119] H. Prothmann, H. Schmeck, S. Tomforde, J. Lyda, J. Hahner, C. Muller-Schloer and J. Branke, “Decentralised Route Guidance in Organic Traffic Control,” in *2011 Fifth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO),*, Ann Arbour, MI, 2011.

[120] M. Cherif, S. Senouci and B. Ducourthial, “A new framework of self-organization of vehicular networks,” in *GIIS '09. Global Information Infrastructure Symposium, 2009.* , Hammemet, 2009.

[121] A. Farahani, H. Sadeghian, M. Abbaspour and E. Nazemi, “A model for autonomic vehicular Ad hoc networks,” *Global Journal on Technology,* vol. 1, pp. 578-584, 2012.

[122] T. Preisler and W. Renz, “Scalability and robustness analysis of a multi-agent based self-healing resource-flow system,” in *2012 Federated Conference on Computer Science and Information Systems (FedCSIS),* , Wroclaw, 2012.

[123] A. Sujil, S. Agrwal and R. Kumar, “Centralized multi-agent self-healing power system with super conducting fault current limiter,” in *2013 IEEE Conference on Information & Communication Technologies (ICT),* , Jeju Island, 2013.

[124] D. Ductegor, “An agent-based wide area protection scheme for self-healing grids,” in *2011 IEEE PES Conference on Innovative Smart Grid Technologies - Middle East (ISGT Middle East),* , Jeddah, 2011.

[125] G. Zhabelova and V. Vyatkin, “Multi-agent Smart Grid Automation Architecture based on IEC 61850/61499 Intelligent Logical Nodes,” *IEEE Transactions on Industrial Electronics,* 2011.

[126] R. Gupta, D. Jha, V. Yadav and S. Kumar, “A multi-agent based self-healing smart grid,” in *2013 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC),*, Kowloon, 2013.

[127] M. Zadeh and M. seyyedi, “A self-healing architecture for web services based on failure prediction and a multi agent system,” in *2011 Fourth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT),*, Stevens Point, WI, 2011.

[128] R. Pegoraro, H. Filho, M. Sacoman and J. Rosario, “A Self-Healing Architecture for Web Service-Based Applications,” in *11th IEEE International Conference on Computational Science and Engineering Workshops, 2008. CSEWORKSHOPS '08.* , San Paulo, 2008.

[129] K. G. K. D. M. J. Riadh Ben Halima, “Providing Predictive Self-Healing for Web Services: A QoS Monitoring and Analysis-based Approach,” *oumal of Information Assurance and Security,* vol. 3, pp. 175-184, 2008.

[130] U. Reiner, C. Elsinger and T. Leibfried, “Distributed self organising Electric Vehicle charge controller system: Peak power demand and grid load reduction with adaptive EV charging stations,” in *2012 IEEE International Electric Vehicle Conference (IEVC),* , Greenville, SC, 2012.

[131] M. Di Bisceglie, C. Galdi, A. Vaccaro and D. Villacci, “Cooperative sensor networks for voltage quality monitoring in smart grids,” in *PowerTech, 2009 IEEE Bucharest*, Bucharest , 2009.

[132] S. Srivastava, S. Suryanarayanan, P. Ribeiro, D. Cartes and M. Stcurer, “A conceptual power quality monitoring technique based on multi-agent systems,” in *Proceedings of the 37th Annual North American Power Symposium, 2005.* , 2005.

[133] H. Liu, Y. Chen, M. C. Chuah and J. Yang, “Towards self-healing smart grid via intelligent local controller switching under jamming,” in *2013 IEEE Conference on Communications and Network Security (CNS)*, National Harbour, MD, 2013.

[134] P. Casey, N. Jaber and K. Tepe, “Design and implementation of a cross-platform sensor network for smart grid transmission line monitoring,” in *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm),* , Brussels, 2011.

[135] C. Frenzel, S. Lohmuller and L. Schmelz, “SON management based on weighted objectives and combined SON Function models,” in *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*, Barcelona, 2014.

[136] R. B. Halima, K. Guennoun, K. Drira and M. Jmaiel, “Providing Predictive Self-Healing for Web Services: A QoS Monitoring and Analysis-based Approach,” *oumal of Information Assurance and Security,* vol. 3, pp. 175-184, 2008.

[137] C. Gao, *PhD Thesis: Voltage Control in Distribution Networks using On-Load Tap Changer Transformers,* Bath: Department of Electronic and Electrical Engineering University of Bath, 2013.

[138] ZigBee Alliance, “ZigBee Smart Energy Profile Specification,” ZigBee Standards Organization: , 2008.

[139] T. R. Burchfield, S. Venkatesan and D. Weiner, “Maximizing Throughput in ZigBee Wireless Networks through Analysis, Simulations and Implementations,” in *Proceedings of the International Workshop on Localized Algorithms and Protocols for Wireless Sensor Networks*, Santa Fe, 2007.

[140] M. Farooq and T. Kunz, “On Determining Bandwidth Usage Threshold to Support Real-Time Multimedia Applications in Wireless Multimedia Sensor Networks,” in *2013 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, Barcelona, 2013.

[141] T. Abdullah, L. Mhamdi, B. Pourebrahimi and K. Bertels, “Resource Discovery with Dynamic Matchmakers in Ad Hoc Grid,” in *ICONS '09. Fourth International Conference on Systems, 2009*, Gosier, Guadeloupe, 2009.

[142] S. Naaz, A. Alam and R. Biswas, “Effect of different defuzzification methods in a fuzzy based load balancing application,” *IJCSI International Journal of Computer Science Issues,* vol. 8, no. 5, pp. 261-267, 2011.

[143] Data Communication Company, “About DCC,” Data Communication Company, [Online]. Available: https://www.smartdcc.co.uk/about-dcc/. [Accessed 19 October 2016].

[144] Smart Energy Code Company, “Smart Energy Code - Update 4.15 (18th August 2016),” Smart Energy Code Company, 2013.

[145] Data Communications Company, “Infrastructure Key Infrastructure (IKI) Certificate Policy (CP),” Data Communications Company, 2015.

[146] O. Block, “Assessing open source software usage in the development of grid control and measurement device software - A suggestion for an easy to use tool promoting communication between developer and legal professional,” in *2012 International Conference on Smart Grid Technology, Economics and Policies (SG-TEP)*, Nuremberg, 2012.

[147] S. Liu, B. Chen, T. Zourntos, D. Kundur and K. Butler-Purry, “A Coordinated Multi-Switch Attack for Cascading Failures in Smart Grid,” *IEEE Transactions on Smart Grid,* vol. 5, no. 3, pp. 1183-1195, 2014.

[148] K. Samarakoon, J. Ekanayake and N. Jenkins, “Reporting Available Demand Response,” *IEEE Transactions on Smart Grid,* vol. 4, no. 4, pp. 1842 - 1851, 2013.

[149] P. Xin, “IEC 61850 Testing and Documentation,” Vaasan ammattikorkeakoulu, University of Applied Sciences: Available Online - https://www.theseus.fi/bitstream/handle/10024/17035/Peng_Xin.pdf?sequence=1, Vaasa, 2010.

[150] P. Diefenderfer and P. Jansson, “Power sensor applications in a load management network for a residential microgrid,” in *2014 IEEE Sensors Applications Symposium (SAS),* , Queenstown, 2014.

[151] S. Hahn and T. Kurner, “Managing and altering mobile radio networks by using SON function performance models,” in *2014 11th International Symposium on Wireless Communications Systems (ISWCS),* , Barcelona, 2014.

[152] H. Yan, Z. Shi and J. Cui, “DBR: depth-based routing for underwater sensor networks,” in *Proc. IFIP-TC6 Networking Conf. on Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, 2008.

[153] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, 2000*, 2000.

[154] G. Smaragdakis, I. Matta and A. Bestavros, “SEP: A Stable Election Protocol for clustered heterogeneous wireless sensor networks,” in *Proceedings of 2nd International Workshop on Sensor and Actor Network Protocols and Applications*, 2004.

[155] J. van der Horst and J. Noble, “Distributed and Centralized Task Allocation: When and Where to Use Them,” in *2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop (SASOW)*, Budapest, 2010.

[156] V. Rigoni, “Deliverable 3.7 “Characterisation of LV Networks",” University of Manchester: Available Online - http://www.enwl.co.uk/docs/default-source/future-low-voltage/university-of-manchester-appendix-j.pdf?sfvrsn=2, Manchester, 2014.