Using Agent-Oriented Neural Networks for Identifying Network Nodes

by

Andrew Scott

A dissertation submitted to the University of Dublin, in partial fulfilment of the requirements for the degree of Master of Science in Mobile & Ubiquitous Computing.

Department of Computer Science and Statistics

The University of Dublin, Trinity College

September 2010

Declaration of originality

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed ____

Date ____

Acknowledgements

I would like to express my great debt of gratitude to my wonderful supervisor Dr. Stefan Weber, whose calm demeanour and generosity with his time made this research a pleasure to undertake. I will miss our great chats and lengthy meetings.

I would also like to thank my amazing family, my girlfriend Róisín and my friends for their tolerance and support throughout my MSc.

Abstract

The future of mobile and ubiquitous computing presents many novel challenges for protocol design. The increasing popularity of dynamic, Mobile Ad-hoc Networks with heterogeneous nodes has meant that major questions have been asked about the suitability of IP addressing as a means of identifying nodes in modern networks.

This research presents a new way of creating identities for communicating devices in modern networks. The novel concept of an Agent-Oriented Neural Network is introduced as a means of evolving a *personality* for a device. The device's evolved *personality* reflects the functional capabilities of the device and its usage patterns.

The personal identity is designed to aid reliable route path selection and to promote respect for other devices in heterogeneous networks. This is achieved by providing information about what a device is capable of handling. Nodes will be able to select next-hop recipients with similar functional capabilities to themselves, knowing that they can reliably forward their load as their *personality* reflects similar traits.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This research is concerned with developing a new method of identifying network nodes with a view to rectify the growing constraints posed by current addressing protocols such as IPv4 and IPv6, which date back as far as 1980.

In this research a solution is presented to address some of the problems that current node identification techniques cause for effective data dissemination in modern networks. The solution discussed in this research is centred on a new means of identifying devices in a more meaningful way. A novel concept called an Agent-Oriented Neural Network (AONN) is introduced, which takes a node's usage patterns by monitoring the activity of hardware modules and evolves a unique *personality* for the node. The *personality* is based on its interactions with its environment, with other nodes and with the devices used, where applicable.

The research undertaken in this dissertation is regarded as a contribution to the work done in the field of content and characteristic-based routing approaches.

1.1 Research Question

The research question that this dissertation addresses is:

Can an identity be built for a network node that provides details about what kind of device it is and how it is used?

1.2 Context

The trends, for the future of the Internet and the future of mobile and ubiquitous computing, show that the traditional methods of identifying network nodes are not suitable for the dynamic nature of the modern topology-free networks, such as Mobile Ad-hoc Networks (MANETs). The TCP/IP stack protocols were designed for, and for the most part still rely on, fixed network structures using static gateways to interconnect networks.

For the majority of node identification techniques, the identity given to a device is bereft of any information about the device itself or what it is capable of. Some modern approaches to routing protocol design observe characteristics of devices or content of the messages nodes are transferring. This approach is taken in order to create a more meaningful method of choosing routing paths for data in a network.

An assumption is made in this research to motivate the design. The assumption is that in a highly dynamic MANET it is more beneficial to know what kind of device your payload is being routed through for reliability and respect for other devices in the network (se *Fig.1*). For instance, it is assumed to be less favourable to route a large video stream through a Bluetooth enabled, low power, sensing module, as its computational ability would not be sufficient to handle such a load. Furthermore, it is assumed to be better practice not to overload devices, of low computational power, with large complex loads that may adversely affect the way they perform the duties they were designed to perform.

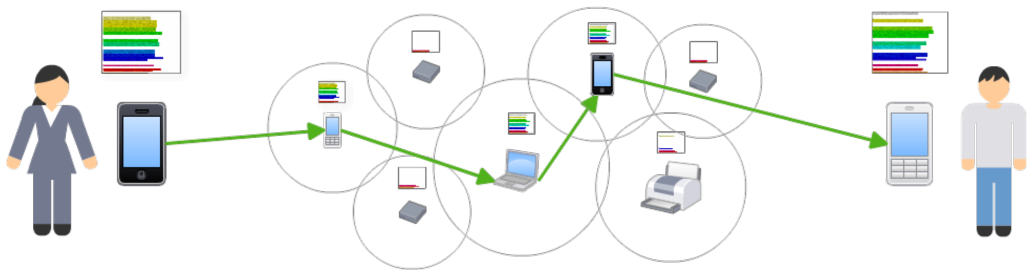


Fig.1. Path through MANET with heterogeneous nodes.

1.3 Aim and Objectives

The aim of this research is to identify the challenges that are introduced by emerging mobile technologies. The research aims to identify the current state-of-the-art and analyse it with a view to its potential limitations. The main goal is to introduce a novel method of identifying network nodes, which are focused primarily on catering for the needs of devices in dynamic topology-free networks such as dynamic MANETs with heterogeneous nodes. Once the solution design is presented to the reader it is the aim of the author to critically evaluate the solution with a comparative study and analysis in order to ascertain whether the solution could be applied in the real world.

The overall objective of this research is to introduce a new method of identifying network nodes, which can help future mobile technologies integrate seamlessly into the established infrastructure. Whether the outcome of the presented solution is successful or not, it is the author's goal to inform the research community of the findings of this dissertation in order to aid meaningful research in the future.

1.4 Motivation

The motivation for this research is the lack of adaptability and suitability of many of the node identification systems currently in place. The aim of this research is to try to offer a better method of identifying network nodes, which reflects the functionality and capability of the device. With a view to the future, it seems imperative to begin taking a new look at what is considered important information for communicating in a network.

This research is motivated also by the required complexity of routing protocols due to the current node identification techniques and by the issue of address space depletion.

1.5 Study Background

There are four main areas of interest in this research the first of which is IP addressing. Since the beginning of the Internet there has been some means of identifying the end nodes of a communication. As the Internet has evolved, technologically as well as in scale, new challenges are arising. We find ourselves at a point of uncertainty, where a new focus on mobile technology is revealing the limitations of the current Internet infrastructure. However, because of how well established the infrastructure is, it is almost unimaginable to consider what would be involved in implementing some new design for it. With this considered, it is not the author's intention to attempt to reinvent the Internet. Instead the solution this research hopes to provide is a new means of identifying network nodes in such a way that allows for less reliance on static addressing schemes. As every node has a number of hardware modules installed, there is the scope for generating an identity for each node based on its own unique usage patterns.

The second area of interest in this research is with regard to Mobile Ad-hoc Networks (MANETs). MANETs are becoming more prominent in the development of the future of mobile computing and with these new networks comes inherent challenges for the legacy protocols still widely in use. Many new approaches have been introduced to help to integrate MANETs into the current infrastructure and new protocols pioneered to best suit the dynamism of these ever-changing networks.

The third area of interest in this research is Characteristic or content-based routing protocols. Characteristic-based routing protocols attempt to remedy the static nature of IP addressing by attaching some metaphor to the nodes in the network and using a characteristic of this metaphor, rather than some static ID, to help improve the freedom of a node. In this research a number of these techniques will be identified to best assess where the author's contribution lies. One of the main challenges for the future of MANETs is to try to either break down the static constraints of the Internet infrastructure – with its minefield of gateways and routers and switches etc. – or improve the current technologies to overcome these constraints.

This research will follow in the vein of Characteristic and or content-based ideology. By creating a *personality* for a device it is possible that routing protocols could utilize this knowledge to improve how they function. When nodes have *personalities* based on the hardware technologies they have installed, it is projected that they will be more meaningfully identified. Furthermore, it is projected that this will aid the discovery of reliable routing paths as a node's suitability for processing or forwarding data packets will be apparent in its *personality.*

The fourth and final area of interest in this research is Pattern generation. For the solution introduced in this research an approach of pattern generation is adopted. The solution described in this research involves intelligently evolving a *personality* from a node's hardware usage data. For this the Agent-Oriented Neural Network (AONN) is introduced, which merges some Multi-Agent Systems (MAS) ideology with Artificial Neural Networks (ANN) to produce a more powerful pattern-generating solution.

In this research the use of ANNs is investigated for pattern generation as they have been utilized to great effect for memory modelling and pattern recognition in the fields of Machine Learning and Connetionist computing [1][2].

A set of Goals, Beliefs and Actions are formulated for each Neuron so that they fire deliberately rather that in accordance with some binary threshold function.

In this regard MASs play two crucial roles: 1) for adding logical reasoning to Neurons; and 2) for gaining insight into Agent behaviours for cooperative versus non-cooperative Agents and deliberative versus reactive or bold Agents [3].

1.6 Dissertation Structure

In Chapter 2 the current state-of-the-art is defined with regard to current node identification techniques, focusing on aspects of mobility.

In Chapter 3 a discussion about the solution design is presented. Both basic and advanced design perspectives are discussed.

Chapter 4 provides a discussion about the solution's implementation design.

Chapter 5 presents a comparative evaluation of the solution with respect to the current state-of-the-art.

Finally, the conclusions drawn from this research are presented in Chapter 6. Some suggestions for future work are offered.

Chapter 2

State Of The Art

In this chapter a literature review is presented which details the current state-of-the-art for node identification.

2.1 The Internet Protocol

Identifying nodes in a network is crucial as it facilitates the routing of packets from a source to a destination. Without node identification, packets may never be delivered to the desired recipient. The IP protocol is a network or Internet layer protocol, which looks after the addressing and forwarding of packets. From the very beginning of networking there has always been some means of nodes identification. However, not all have been as suitable for the growing Internet as IP addressing. IP addressing was designed for identifying many millions of nodes within local networks, which are topologically disparate from one another. Fundamentally IP addressing is a static addressing protocol, which is infrastructure and location dependent. It also relies on static gateways to assign network addresses. Many suggested solutions such as SIP in Driessen et al [4] have failed, however they have paved the way for pioneering thought for approaching the Internet protocol.

2.1.1 History

One of the earliest recorded descriptions of a network-based social interaction was by J.C.R. Licklider of MIT in a series of memos from August 1962 detailing his "Galactic Network" concept [5]. His vision was a globally interconnected network of computers, which could facilitate fast data access and programs from anywhere in the world. Fundamentally, Licklider envisioned the modern Internet.

Following Licklider's advice Leonard Kleinrock convinced Lawrence Roberts that a global internetwork was theoretically feasible, using packet communications rather than circuits. This was a major innovation in the field of data networks and began a new era of computer-based networking. Another major consideration in this was to get the computers to communicate with one another. In 1965 the first connection was made between two computers using this new computer networking approach, the TX-2 computer in MIT and the Q-32 in California using a low speed dial-up telephone line. This was marked as the very first wide-area computer network [6].

In 1966 Roberts began developing the computer network concept at DARPA where he pioneered his plan for "ARPANET", the first internetwork [7]. For the first time the term "packet" was used to describe data segments and a proposed line speed was suggested as 50kbps for the ARPANET.

By August 1968, Roberts along with other DARPA researchers had outlined the main structure and specifications for the ARPANET. However, one of the main areas of interest remained in question, the packet switches called Interface Message Processors (IMP).

The Internet Protocol was first described by Cerf and Kahn [8], as "a protocol that supports the sharing of resources that exist in different packet switching networks". The protocol outlined in this work was designed to provide for a number of challenges inherent in packet-switching networks such as, varying packet sizes, packet transmission failures, packet sequencing, flow control, end-to-end error checking, and logical process-to-process connections. The authors introduce the concept of Gateways as an interface between networks, which allowed for packets to be transmitted from one network to another. The introduction of the Gateway here adds an addressing complexity to the routing of packets from a source to a destination, as with Gateways it is possible to transfer packets through networks of different types. The Gateways convert packets into the format of the network they interface and thus the addition of an "internetworking header" in the packet's prefix. The authors outline the need for a uniform address space common to all networks in order for there to be cooperation between the TCPs at each "host" and to allow for packet fragmentation and re-assembly to happen successfully. While introducing the notion of ports the authors also speculate the first IP address called a TCD address, which is a 24-bit address for TCP to TCP connection.

By 1969 four host computers were connected together to form the initial ARPANET, which formed the first internetworked packet switching network and thus the foundation for the Internet [9]. In the same year the first Request For Comments (RFC) was published by Internet Engineering Task Force (IETF) on the ARPANET as a means of gathering feedback on ideas and questions asked about ARPANET. RFC 1 [11] was the first RFC published on APRPANET's Host Software. It outlines some fundamental attributes of the IMPs and Hosts on the ARPANET. At this early stage messages were send with a 16-bit header containing a 5-bit destination address, an 8-bit link header, a 1-bit trace bit and 2 unused spare bits. In these first stages of the Internet a 5-bit address was sufficient for identifying an end node due to the small number of nodes on the internetwork.

In 1972 Robert E. Kahn outlined some rules for a new protocol, which with the help of Vinton Cerf in 1973, became the Transmission Control Protocol/Internet Protocol (TCP/IP) [9]. This new protocol effectively replaced the old Network Control Protocol (NCP) used prior to this. With it was brought end-to-end host error control and more reliable packet transmission. Thus supporting a more communication-oriented, open-architecture network. For this protocol a 32-bit IP address was sufficient for the vision of ARPANET by its creators as a national infrastructure of relatively small numbers of nodes. The first 8-bits were dedicated to identifying the network and the remaining 24-bits for identifying the host within the network.

At the same time Xerox PARC was developing Ethernet and local area netwroks. However, engineers at DARPA did not consider internetworks of more than 256 individual networks – evident in their modest Address space provided in the early TCP/IP.

In [10] a series of APRPANET protocols were put to the test to evaluate their performance with respect to packet throughput over long distances. The protocols, which were a development of some of the ideas expressed in [8], are shown to perform quite well under the described circumstances. As their addressing mechanism was much the same as that described by Cerf and Kahn [8], it was shown that the main concern for the early internetworks was not with addressing but with flow control and message processing.

When efforts were made to develop more advanced network applications TCP/IP was found to hinder the performance of some of these applications where some packet loss should not be corrected. This led to the split of TCP and IP into two separate, more specific, protocols. User Datagram Protocol (UDP) was developed to bridge the gap and IP's role was confined to simply addressing and forwarding [9].

As Xerox PARC were focused on developing personal workstations such as the Xerox Alto, David Clark of MIT set out to simplify TCP to run on these new machines. This proved that PCs and workstations could also join the Internet and with the early development of email in DARPA there was a new surge interest in a more inclusive Internet.

Through the 1980s, the Ethernet innovations at Xerox PARC, led by Bob Metcalfe, helped to develop the Internet and its protocols for use with much larger numbers of processing nodes than had been considered by the ARPANET. In RFC 760 (1980) [11] the IPv4 address was redefined as a 32-bit ID, 8-bits as the network number and 24 bits for the local address, which were assigned by the local network. This would allow for a single host to appear as multiple hosts. However Ethernet would soon adopt a 48-bit Media Access Control (MAC) address instead to cater for the new volumes and types of traffic for the new Ethernet networks.

The development of the IP through the 1980s was very much concentrated on the physical location of networks and nodes and the infrastructure of the whole Internet was formed on the assumption of static nodes. With the introduction of Domain Name Servers (DNSs), Interior Gateway Protocols (IGPs) and Exterior Gateway Protocols (EGPs) the IP address evolved into an identifier, which indicated a physical location inside a hierarchy of spatially specific gateways and routers.

With the privatization of the Internet infrastructure came a great competitive upsurge in technological development for company gain. This drove the scale of the Internet through the 1990s and pushed technological advances. The more nodes coming on line the more the technology was refined and the more the IP address space was being exhausted.

In 1998 with the Internet expanding rapidly the IETF recognized the depletion of the IPv4 address space. As a solution they devised an expanded namespace called IPv6 (RFC 2460) [11]. IPv6 was defined as a 128-bit address allowing 2^{128} possible addresses. While this measure solves the current problem of available IP addresses it still assumes that a node is connecting to a static access point to a local network somewhere.

2.1.2 Limitations of IP

IP's limitations can be attributed to some basic characteristics i.e. in order to send a packet over the Internet a node must have an IP address. A node's IP address is an indication of the computer's physical location. The TCP/IP protocol routes packets from a source to a destination using an IP address. These factors produce some major limitations for IP. Such as if a mobile node moves between network access points without changing its IP address, the routing is lost and if a node changes its IP address its connection to the network is lost. Thus, with IP there is very poor crossover between Wifi networks. Packets will be lost either, when the connection fails or the route fails which means the network is unreliable [12][13].

For proper mobility the wireless Internet will have to provide all of the services available on the Internet and should be reliable. It should provide reasonable throughput both indoors and outdoors for both mobile and stationary nodes. It should use energy efficiently as most devices run on batteries and should scale up to support millions of active devices in a single metropolitan area [13][14].

2.1.3 Mobile IP

Mobile IP is a new set of protocols created for mobile computing. It is an improvement of IPv6, which enables nodes to continuously receive data packets regardless of their access point to the Internet. Mobile nodes can still maintain communication with others when passing between access points to the Internet, using the same IP address. However, mobile IP still cannot facilitate smooth network handover or fast mobility [13].

Packets are routed end-to-end from a source node to a destination node using IP. This is facilitated by forwarding packets from incoming network interfaces to outbound interfaces. The routing of these packets is done according to a routing table, which maintains all next hop information about each destination IP address. The routing table's information is based on the number of networks a particular IP address is connected to. In the IP address the bits that contain information about the node's point of attachment to the network are generally masked. From this the network number is derived.

In order to sustain the existing transport layer connections a node must keep its IP address the same while travelling from place to place. In TCP, connections are indexed using a quadruplet header that contains the IP addresses and port numbers of both connection endpoints. If any of these four numbers are changed the connection cannot be established properly and will be lost. However, in order for packets to be correctly delivered to a mobile node's point of attachment, it requires the correct network number from the node's IP address, which changes for each new point of attachment. If the routing is to be changed then a new IP address must be assigned to the new route, as the end points of attachment will change.

Mobile IP aims to solve the problem of IP re-assignment by issuing each mobile node with two IP addresses: a home address and a care-of address. The home address is static and is used to identify a node for TCP connections. The care-of address is dynamic and is assigned each time a new point of attachment is established. This address can be considered as mobile node's topologically significant address – it contains the network number, which identifies the node's point of attachment location in the network topology. [12]

2.2 Mobile Ad-hoc Networks

In this section I will be looking at the nature of MANETs and their implementation of Transport protocols for data transmission in networks with no topology.

Mobile Ad-Hoc Networks are a type of wireless ad hoc network. They typically exist in a routable networking environment on top of a Link Layer ad hoc network. Generally, each device in a MANET is free to move around from point to point connecting to many different devices and access points. For these behaviours a MANET's topology should be a self-organizing and each node should participate in forward packets from other nodes, which may be unrelated. One of the main challenges for building a network like this to ensure that each node can maintain continuous accurate information necessary for routing traffic. MANETs can operate independently or can be connected to the larger Internet [14].

In the last decade there has been a great rise in the number of mobile devices such as laptops and other portable 802.11 powered devices. This has impacted upon the type of network infrastructure needed to serve such devices. MANETs have become an important area of research as a result. There has been a great focus on new protocols and their ability to accommodate an ever-changing network topology.

Currently there are three major classes of MANET: Internet Based Mobile Ad-hoc Networks (IMANET), Vehicular Ad Hoc Networks (VANET) and Intelligent vehicular ad hoc networks (InVANET). IMANET is the most significant of the mobile ad-hoc networks as they are constituted of mobile nodes inter connecting and also connecting directly or indirectly to fixed Internet-gateway nodes. VANETs are comprised of many vehicles interconnecting and connecting with some roadside source. Stucturally and idealistically IMANET and VANETs are quite close. However, the mobility of their respective nodes is very different. Fundamentally, the challenges for fast, reliable packet delivery are the same. However, for VANETs one of the greatest challenges is fast reliable route discovery and establishment of connections.

MANETs have some unique characteristics, which make them inherently more challenging than other network types, such as broadcasting communication, path loss, fading, interference, Doppler shift, transmission rate constraints, and highly frequent routing changes. [14]

2.2.1 Challenges

In this section I will discuss the challenges that MANETs face with the current Internet infrastructure and the challenges faced by a network with such topology-independent characteristics.

There are many technical and research challenges associated with MANETs, which need to be addressed for the successful function of the network. MANETs and their inherent architectural characteristics have many benefits for the intercommunication of modern devices, such as self-reconfiguration and the ability to adapt to certain mobile characteristics (i.e. traffic distributions, transmission conditions, power and load balancing. Although these are benefits to the overall freedom and variety of nodes they also pose some difficult challenges due to their unpredictability. Thus, designing systems on top of these networks can be very tricky. Furthermore, to establish a system of identifying nodes and routing packets with some sort of reliability and robustness is an essential but difficult challenge to overcome. Already some developments in the area have attempted to alleviate some of the challenges born from MANETs unpredictability, such as, distributed MAC and dynamic routing, Wireless Service Location Protocol, Wireless Dynamic Host Configuration Protocol, distributed admission call control, and quality-of-service (QoS)–based routing technique. [15]

2.3 Characteristic-based Routing

Characteristic-based routing protocols attempt to address the limitations of IP addressing. These limitations adversely affect the freedom of the connectivity that modern Internet ready devices should enjoy. Many new ideas have surfaced in response to the needs of modern devices in topology-free networks. One of the most prominent fields of thought is centred on the concept of routing network traffic using some characteristic of a node instead of a static number.

An approach has been developed within the Distributed System Group (DSG) in Trinity College Dublin whereby a network node advertises certain characteristics attributed to themselves such as, an ability to act as a gateway to a wired network etc. Information is propagated through the network in a way similar to gossiping. This approach is designed as a replacement for IP addressing at the network layer [15].

Traffic in a MANET is typically routed through the network from a source to a destination following this characteristic. In [15] packets are picked up by neighbouring nodes and forwarded to nodes with similar characteristics. Characteristics are distributed through the network in the same way that water flows from a spring. The process of packets being delivered resembles, "the following of the stream of water upwards towards the spring".

2.3.1 Discussion About Content-based Routing Strategies

In [16] the authors outline a content-based communication infrastructure for the first time, which marked the beginning of a new service model targeting MANETs. They define their model, not as a replacement for IP but rather, an infrastructure to facilitate interfacing with a publish/subscribe middleware service. The authors outline the architectural criteria for the network model, which is designed to incorporate reliability, security and performance.

The model the authors describe is not intended to replace IP or interfere with any network layer protocols. Rather the model is based on standard physical network architecture where physical components such as routes and hosts are considered and nodes in a graph and their immediate connection links as the arcs between the nodes. Links are assumed to be bi-directional and thus the model is considered to be a non-directed graph. With this graphical model view the authors define their service model to vary substantially from traditional unicast and multicast networks. Instead of datagram addresses being used *r-predicates* and *s-predicates* are used. These define a datagram that a node intends to receive or send respectively. Datagram models and Predicate models outline format specifications for datagrams and how nodes intend to send or receive them. A router in the network stores a routing table based on the graphical model of the network, and forwarding of packets is done based on this routing table. The router maintains and updates its routing table by keeping a register of predicates for adjacent nodes. The forwarding table is used to disseminate datagrams along possible optimal routes.

The advantage of this network model is that it sets up a framework which facilitates ad-hoc networking because nodes in the network simply have to use predicates to signal their intention to participate in data dissemination within a particular network.

This introduction to content-based networking spurred much development in similar ad-hoc routing focused approaches, such as that of the scalable protocol for content-based routing overlay networks by [17] and the first proper implementation of a forwarding algorithm for content-based networks is described by [18]. Both publications extend the model defined by [16].

In [18] the same authors as [16] extend their original model to include a functioning forwarding algorithm. The algorithm is based on content filtering for text documents and used these to handle message predicates. Their initial design yielded some reasonable results but as the authors themselves concede, this is the first attempt to tackle a complex problem.

In [17] the authors outline a protocol called XRoute, which they demonstrate to implement a very convincing routing scheme. It appears to optimize bandwidth and network use by minimizing the size of routing tables being maintained in the system using tree structures. The authors demonstrate a good scalable solution, which can facilitate large numbers of nodes using their XTRIE filtering algorithms running on the application layer of a router.

The authors provided some good optimizations for content-based networking, which could be used to great effect for MANETs. If each node had the potential to act as a router then this could be a very powerful system. Perhaps with a token system an application layer algorithm could judge when a particular network was becoming saturated and then another node could be activated as a router – maybe on the periphery of the network in order to extend the functional limits of the network.

It is clear how a content-based model might facilitate the use of a *personality* described in this research. A node could register its *personality* with a router and an optimized routing path could be devised based on the *personality* as the type of device that is connected is revealed in its *personality*. Possibly one of the most valuable contributions of content-based networking is that it can support the integration of other routing schema based on a different kind of predicate. Predicates could be defined and altered to be almost anything and some of the existing forwarding algorithms and routing schemes could be adapted to the new content or characteristic with relative ease.

One such adaptable routing scheme is outlined in [19]. A "push" and "pull" type mechanism is used with a broadcast protocol in order to propagate route information. In the case of this particular scheme, message predicates are processed using a matching algorithm. This is a potentially costly solution for the networks of limited processing nodes, such as WSNs. However, it seems like a viable possibility for less resource constrained networks.

Naturally, it would seem that this kind of network is more suitable for the types of networks we come to imagine in the future. We imagine autonomously maintained networks where each node actively routes packets for the network and perhaps nodes become delegated as routers as capacity dictates. Perhaps the rise in popularity in WiMax networks and the move to an all IP-based, packet switched, cell network technology will mean that soon there will be ubiquitous access to the Internet without tricky provider restrictions and costly handover mechanisms.

In [20] the authors evaluate the existing content-based routing protocols and critically analyse their suitability for MANETs. They established that while protocols mostly of provided strengths in some facets such as reliability and fault-tolerance, there was a trade-off for transmission speed. They allude to the fact that it is very difficult to achieve a good balance of all features of the protocols. However, focus on protocols like FT-CBR for highly dynamic network topologies appears to be the favoured choice when considering the current commercial trends in mobile device sales.

Chapter 3

Solution Design

In this chapter the design of the AONN is discussed. Firstly, the basic design overview is presented and the fundamental concepts and terminology for the solution are introduced. A high-level design view is given followed by a more detailed low-level description of the solution design.

3.1 Overview

An Agent-Orient Neural Network is used in this research to generate patterns for network nodes, which represent the device's *personality.* This *personality* is evolved on the device and it reflects the way the device is used in its environment.

The concept of a *personality* for a device could help to identify it better in a network. In the current design for most networks and inter-networks, a system of fixed switches and routers helps to divide collections of devices into geographically significant groupings (networks). Within these networks all devices have IP addresses assigned to them by the network's router. However, many worldwide sales statistics released over the past 5 years from sources such as Gartner [21] and the IDC [22] have shown a significant increase in the numbers of mobile devices entering into wireless network infrastructures around the world. For the majority of devices connecting in these networks IP (802.1x networks) and IMSI (cellular networks) addressing systems are implemented to identify unique devices. For these systems a number is assigned to a device, which provides minimal geographical location information about the device. With 3GPP LTE advanced beginning to roll out across the world the fourth generation of mobile computing will see a huge increase in the demand for IP addresses. With 4.7 billion mobile subscriptions worldwide [23] in 2009 by 2012 it is likely that most of these devices will be using IP packet switching as part of 4th generation telecommunications, and thus massively increasing the load on an already heavily-laden addressing system.

With an AONN-based system a device is uniquely identifiable via a *personality* called an EPID (Evolved Personal Identity). Data is fed into the Agents on the input layer of the AONN. Agents process the information they receive and make a decision on when to fire based on their interaction with other agents and their environment.

Hardware usage is monitored by the AONN such that any activity carried out by a user (where applicable) is reflected in the *personality* of the device. Over time the regular use of a particular feature of a device will strengthen the part of the device's *personality* corresponding to that regularly used feature. For instance if I use my Smartphone everyday to get my location via GPS, I use Wifi regularly to check my emails and I ring my classmates everyday, then GPS, Wifi and Cell Radios will define my particular device's *personality* – thus, reflecting my own. In this way any device in a network can be characterised by its owner and with standardised *personality* evolution each device can quickly establish what every other device is capable of in its transmission range.

This knowledge of other device's capabilities is the key to its preferential routing potential. The assumption is that if two devices are communicating in a highly dynamic ad-hoc environment, full of heterogeneous nodes, then in order to reliably route packets it is more favourable to do so via nodes that are familiar to both end nodes. For example if a mechanical Engineer want to pass information to another mechanical engineer (s)he is not likely to relay the message via a botanist unless the botanist is well read in mechanical engineering. Likewise, in a dynamic ad-hoc network with heterogeneous nodes it is assumed to be better that a Smartphone routes its data through a device similar in computational ability to itself or the destination node. If we consider the scenario of a smart building where there may be thousands of sensors, laptops, PDAs, Smartphones etc. it is better that a laptop which consistently transmits large packets does not utilise a battery-powered sensor node on its routing path as it will adversely affect the performance of the sensor node.

The ability to generate a meaningful identity for a device, which can be generated and utilised quickly and efficiently, is one of the main design concerns of this research.

There are three main motivations behind this design:

1. To help to route data more efficiently in dynamic ad-hoc networks with heterogeneous nodes.
2. To provide a scalable node identification system which will not deplete.
3. Ease the amount of processing required at router and switches.

3.2 Terminology

- ***Agent-Oriented Neural Network (AONN):*** This term refers to the software, which processes the hardware data. It is responsible for generating a *personality* for a device. The AONN is comprised of Agents interconnected in an ANN structure.

- ***Evolved Personal Identity (EPID):*** The EPID is the *personality,* which is evolved using the AONN.

- ***Neural Agent or Neuron:*** A Neural Agent or Neuron in the context of the AONN refers to a computational unit in the network where an Agent is implemented. It is conceptually synonymous with a Binary threshold Neuron in a standard neural network.

- ***Agent:*** An Agent in an AONN is a partially autonomous software agent, which replaces the traditional binary threshold function in a neural network. Agents adhere to the Multi-Agent System view of Agenthood expressed by Yoav Shoham [24].

- ***Synapse:*** In the context of an AONN, Synapses connect Agents of different layers to one another much like Synapses in a Neural Network.

- ***Channel:*** A Channel refers to a conceptual medium through which hardware data is fed into a subset of Neurons on the Input layer of the AONN. Each Channel corresponds to a particular hardware module on the device. The hardware modules assigned to a Channel are standardised for all devices.

- ***Channel Band:*** A Band in the context of an AONN refers to all active Neural Agents from the input layer up to any subsequent layer within a certain range dictated by the Channel. For instance, if Channel 1 feeds data in to Neurons 0 – 3 on the input layer then only Neurons 0 – 3 on all subsequent layers will be included in that Channel's Band.

- ***Channel Set:*** Channel sets refer to the groupings of Channels into three groups. The most common hardware modules for mobile communication devices are in *Set 1* and less common hardware is handled in *Set 2* and *Set 3*.

- ***Snapshot:*** Snapshots are momentary EPID states captured for use in device communication or identification.

- ***Basetime:*** This is a symbolic start time generated on a test device, which is used by the Neural Agents as a reference point. It is created the first time the AONN starts and provides context for the Agents.

3.3 High-Level Design View

In the following section there is a high-level description of the AONN. This incorporates the basic design of the AONN's structure and the overall concepts of the design.

3.3.1 Basic design

The AONN is based on the structure of a Multi-Layer Perceptron (MLP). It is based on a feedforward Neural Network (ffNN) model, which maps input data from the input layer of Neurons onto the Neurons on the output layer. The AONN is comprised of multiple layers of nodes connected in a directed graph. Unlike the traditional MLP, which is fully connected across all layers (see *Fig.2a*), the AONN is fully connected in bands across the network (see *Fig.2b*).

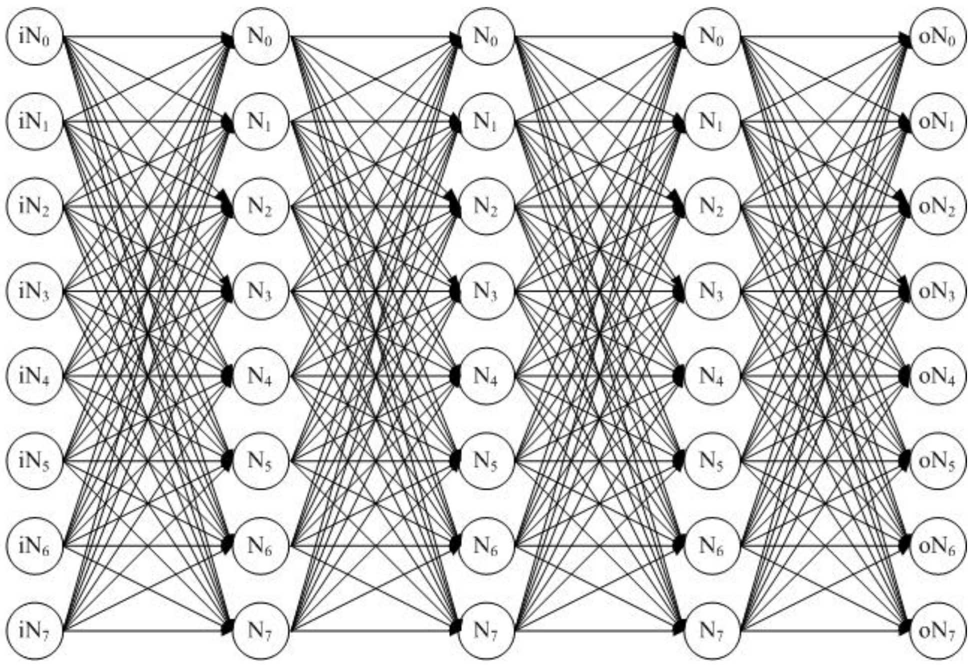


Fig.2a. Multi-layer Perceptron connectivity.

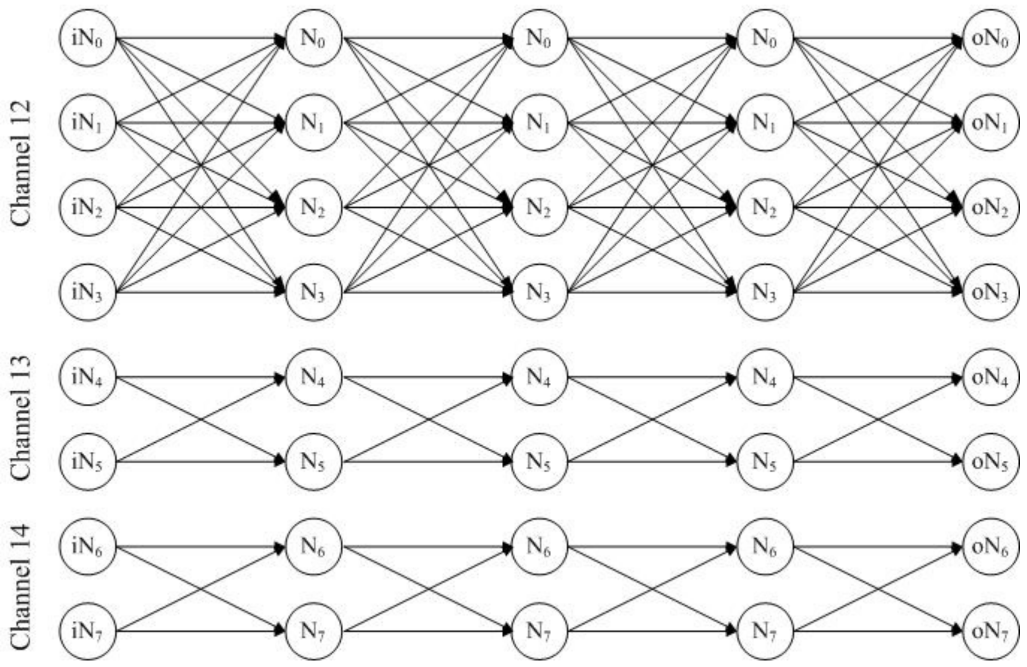


Fig.2b. Agent Oriented Neural Network connectivity.

Each of these bands is fed by a Channel, which corresponds to a specific hardware module on a device. Each of the nodes in the network is a Neural Agent analogous to the processing elements found in an ffNN.

Network Structure

The AONN was initially designed to comprise 64 layers each containing 64 Neural Agents. There is an input layer, L_i, an output layer, L_o, and L_{o-1} – L_{i+1} hidden layers. Unlike an MLP with nonlinear activation functions as computational units, AONN Neurons have the facility to deliberate over when they fire according to a set of beliefs, desires, and intentions.

The Neurons on the input layer are considered to be computational units with the same functionality as Neural Agents in the hidden layers in the network. Input is received at L_o via Channels and each Neural Agent fires a weight value, which is received as input in the Neurons on the next consecutive layer. The media, which carry these weights, are the Synapse.

Channels

In the initial design of the AONN, with 64 x 64 Neurons, 22 Channels were used to feed data into the input Neurons. Each Channel is assigned a particular hardware module to monitor continuously for changes in state. When the hardware is seen to be in use the Channel returns a 1 to the corresponding input Neurons it feeds. The Neural Agents process the data and fire only to Neurons in the next layer, which correspond to the firing Neuron's Channel. In this way the data for hardware modules is processed in bands.

Based on the appearance of current mobile communication device specification trends, a list of the most common technologies to feature in such a device has been compiled. A preliminary standardised list has been set for the designation of hardware modules to Channels for the purposes of this research (see *Fig.3*).

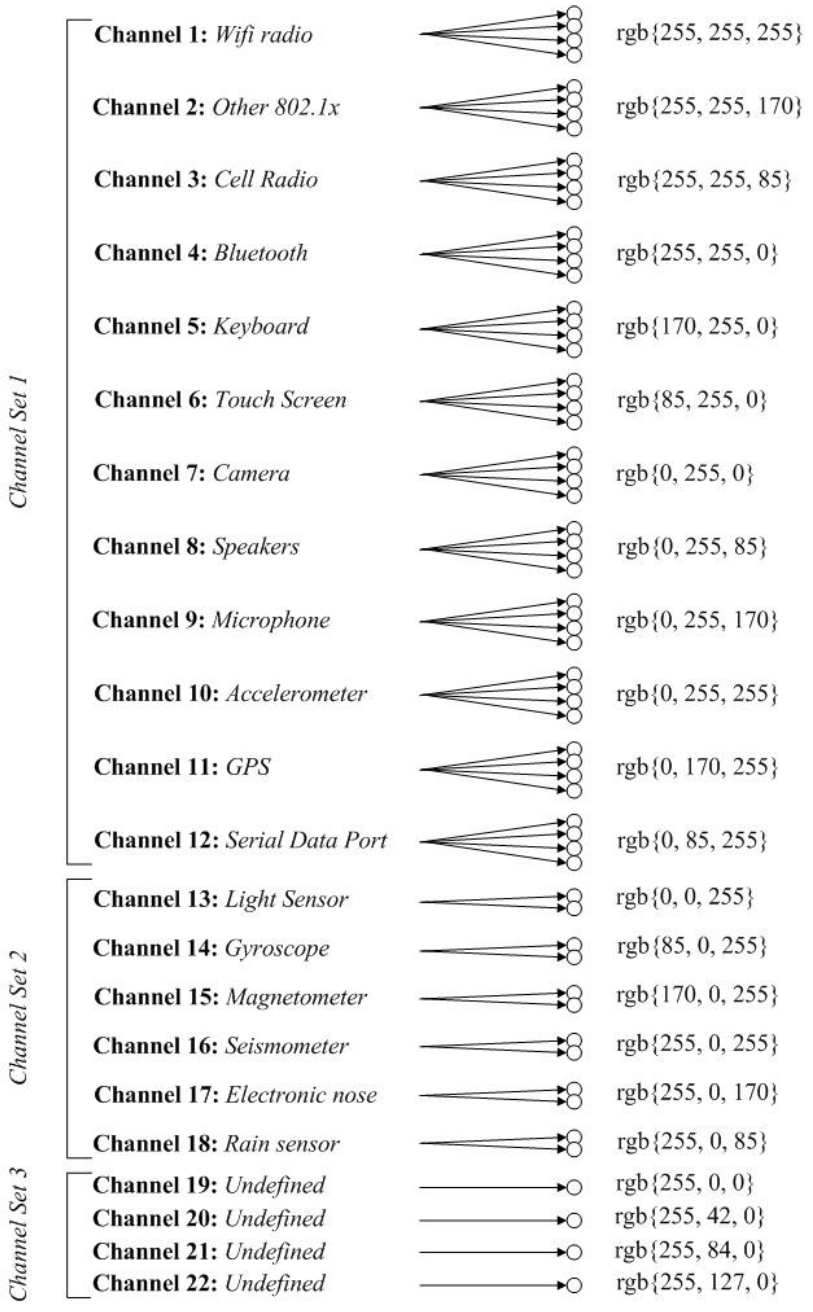


Fig.3. Channel list with assigned technologies and associated colours.

Technologies are assigned to Channels, depending on which appear to be most commonly featured in modern mobile electronic communication devices. Those technologies featured in Channel Set 1 are seen to be the most commonly occurring hardware technologies, whereas those in Channel Set 2 are considered to be less common. Channel Set 3 is designated to other technologies which a vendor or private network may decide to assign to their devices.

EPID

The EPID is evolved over time on a device based on the frequency of use of its hardware modules. If a device is designed to be operated by a user then the user's patterns of operation are reflected in the EPID. Thus, the device's EPID takes on the *personality* of the user.

Scenario

It is possible, at a high level, to establish what sort of device is being identified, by analysing which bands are active in the EPID. For example (see *Fig.4*) device *A* is a Smartphone and it wants to send an important video to device *B*. Device *B* is in an area of healthy cellular radio coverage whereas device *A* is in an underground Metro station with hundreds of people and cannot establish a connection with a cell tower. It sends out a search packet to try to find suitable devices in its environment through which to route its data packets.

Within range of device *A*'s Bluetooth radio is a number of devices with Bluetooth capability. Two of the devices are basic embedded sensor nodes with low computational power. Their EPIDs are very different to device *A*'s. One device within range has a very similar EPID to device *A* but has no access direct point within range. Device *A* sends its EPID to the suitable device, which in turn sends out a search packet and so on returning only a route chosen through nodes with similar EPIDs.

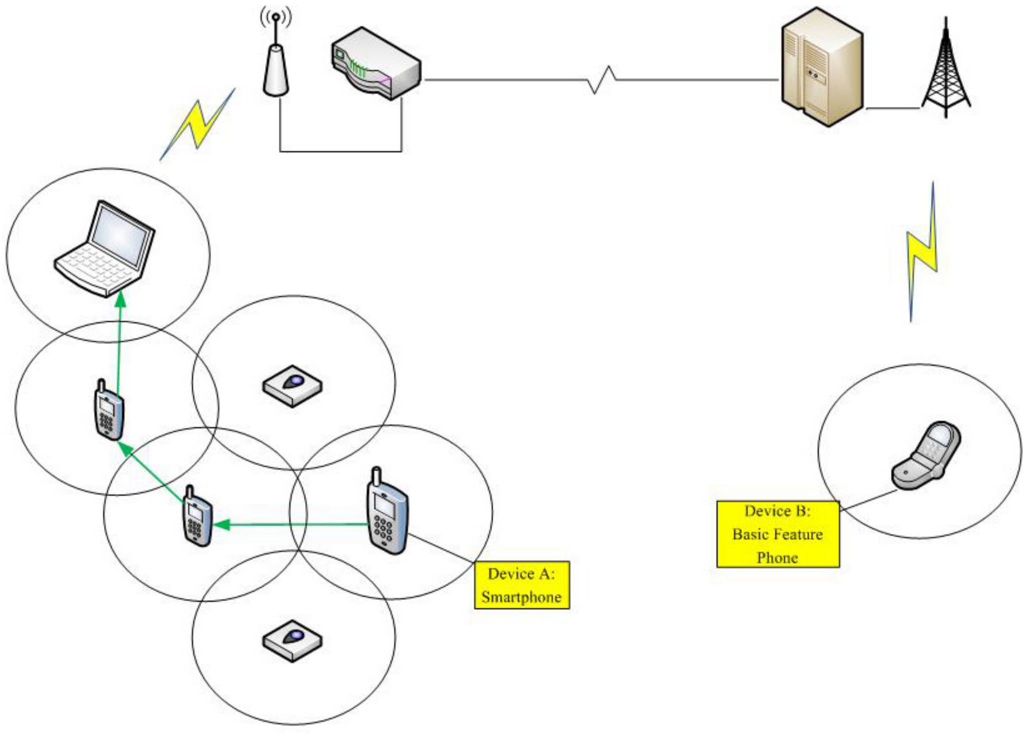


Fig.4. Example scenario of preferential routing based on EPIDs.

As such an EPID is designed to aid existing routing protocols by choosing routes through familiar devices, when using EPIDs to route traffic in a highly dynamic MANET with heterogeneous nodes, it is assumed to be beneficial to know what sort of devices share a network space. If devices know each other's capabilities it is assumed that this knowledge can help them to reliably and more efficiently route traffic.

3.4 Low-Level Design View

In this section a more advanced insight into the design and function of the AONN is given. A discussion about the design choices and iterations is also included.

3.4.1 AONN Advanced Design

Initial Design

From the initial design phase it was decided to use a fully connected network (see *Fig.2a*) with a size of 64 x 64 Neurons, as this would provide a highly detailed view of the devices' use patterns. In the early stages the network was fully connected (see *Fig.2a*) and weights were stored on Synapses, as is the case with standard MLPs. With this design it meant that 63 x 4096 separate data instances were used to identify a device. A weight value in the range of 0 - 7 was stored on each Synapse meaning that there was a potential for 8^{258048} identities to be generated.

Each Neuron was initially given a basic set of knowledge about its environment: Basetime, current time and location in the network. This context is important for the Neural Agents as their decision to fire depends on how much time has elapsed in its lifespan and also since the last time it fired. Furthermore, knowledge of where exactly in the network a Neural Agent is located is essential for making their decision to fire.

The decision to use a Multi-Agent approach was to investigate the power of computation that such a design could achieve. For standard ANNs such as MLPs each computational unit (Neuron) is comprised of a binary threshold function, such as:

$$y = \sum_{i=1}^{n} w_i x_i \qquad y = \begin{Bmatrix} 1 \text{ if } y \geq \theta \\ 0 \text{ if } y \leq \theta \end{Bmatrix}$$

where $\boldsymbol{y}$ is the output fired by the Neuron. This step function sums all weight values, $\boldsymbol{w}$, it receives as input and fires when a threshold $\boldsymbol{\theta}$ is reached. While this method provides a means by which basic simulated memory and recognition can be reproduced it is still very limited and not very sophisticated.

Initially the goal was to attempt to create an ANN where Neurons were more sophisticated and could operate autonomously without the requirement for training or supervision. The benefit of this would be to add a more deliberated approach to the evolution of an EPID as each Agent makes a calculated decision to fire.

Neural Agents were designed to hold beliefs, desires and intentions in accordance with the BDI model described by Anand and Georgeff [25]. For this solution these beliefs, desires and intensions are defined as follows:

- ***Belief:*** The belief set for the agent includes updated information about the current system time and information about when the Agent last fired. Also the Agents' belief-set includes knowledge about where it is in located in the AONN.

- ***Desire:*** Agents' desire is to fire a value to the next layer. An Agent's goals, desires and intentions are closely related. In this case the Agent only has one desire and that is to fire.
- ***Intention:*** The Agents' intentions are defined such that the agent will fire once it has received a set number of input messages from the preceding layer. It intends to fire once a set time has elapsed. The set time corresponds to the layer the Agent is in. The closer it is to the output layer the greater the amount of time that must elapse between firings.

The Neural Agents were designed to function as depicted in the structure in *Fig.5* below.

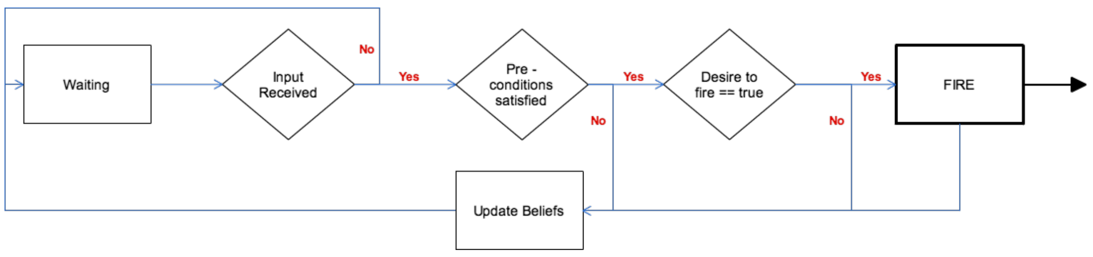


Fig.5. Agent Architecture design.

The Basetime set at the first launch of the AONN denotes the beginning of an Agent's life. This is used as a reference for all decisions to fire during the evolution period of the EPID. Time segments are distributed over the layers, the sum of which amount to a period of $t_e \cdot 8^{64}$, where t_e is the time segment to be elapsed. The length of time segments increases linearly the closer an Agents is to the output layer. The distribution of time segments can be represented by the graph in *Fig.6*. When an Agent's allotted time elapses it fires with a value based on how much input it has received since the last time it fired. A value of between 0 - 7 is then stored on the Synapse and for human readability was assigned a colour corresponding to its weight.

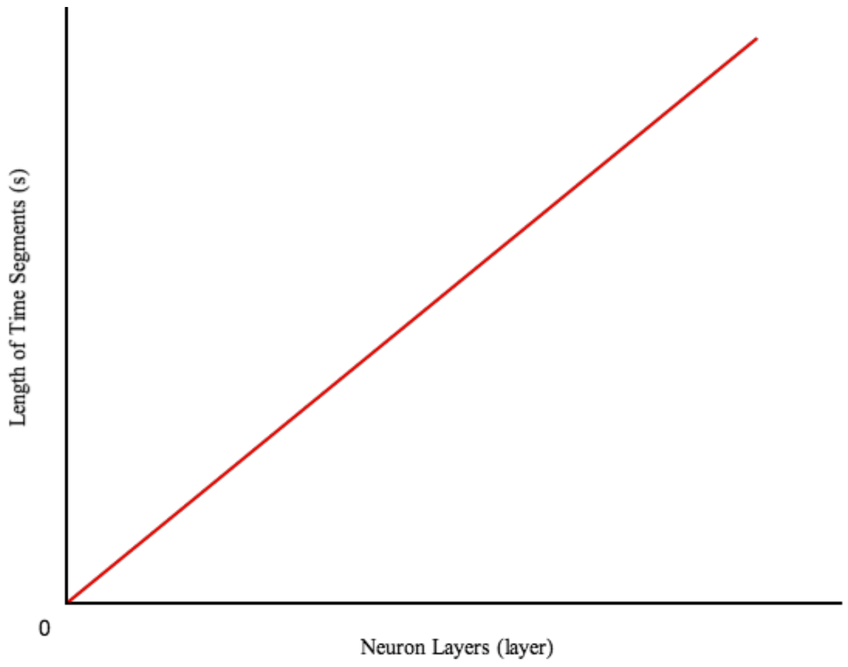


Fig.6. Linear distribution of time segments over AONN layers.

First Design Revision

Due to the amount of computational expense incurred by this design and the time it took to process all of the information the design was deemed unfeasible and impractical and as such a decision to take a new design approach was made.

As each Agent is, by nature, designed to be autonomous it meant that when it came to the implementation stage that the ample test equipment, used for developing the prototype AONN, struggled to manage the computational load.

The first revision of the design involved changing where the weight values were stored. It was decided, that in order to reduced the amount of memory used by the AONN, weight values would be stored at Neural Agents rather than on Synapses. This meant that the number of possible identities the AONN could generate would be reduced to 8^{4096}, which is still a very large number.

It was easier to manage the AONN with this revision as the data returned by the Network was more closely related to its structure. This meant that rendering a humanly readable version of the EPID was possible (see *Fig.7*).

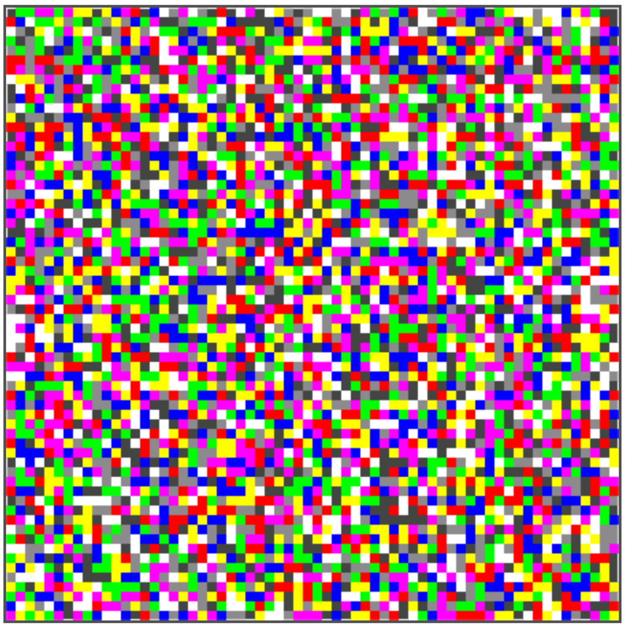


Fig.7. EPID in design phase 2.

Second Design Revision

The second revision was to redesign how the Synapses connected the Neurons. It was decided that the Channels concept should be expanded to a concept of Bands across the EPID. This meant that the Neural Agents fed by a particular Channel would fire only to an equivalent number of

Neurons on the next layer within the same range. *Fig.8* below shows a Channel within Channel Set 1 where a hardware module feeds a range of 4 Neurons at the input layer – **iN$_0$** to **iN$_3$**. The Neurons in this range fire only to Neurons in the same range, on the next layer – **N$_0$** to **N$_3$**.

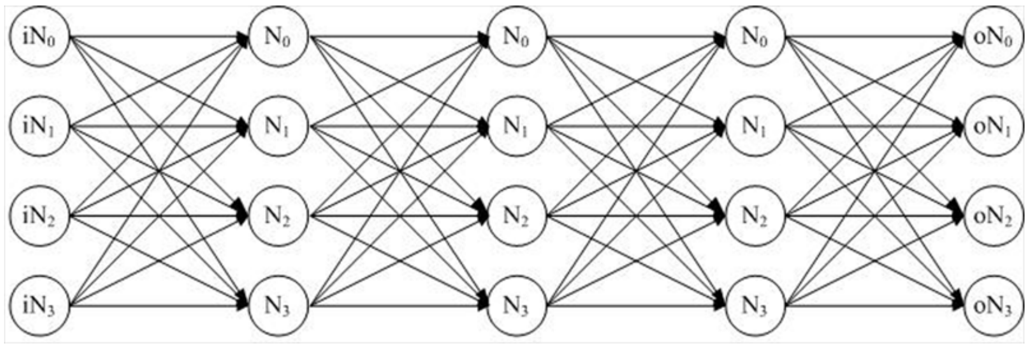


Fig.8. A Band created by segmenting Neurons across the layers.

Instead of having a colour representing a weight value on a Neuron, with this design revision the colour was assigned to a particular Channel. As the Channel was assigned a particular colour the Band associated with the Channel thus inherited the colour. Neural Agents were redesigned also to be inactive until such time as the Neurons on the previous layer fired weight values to it. This meant that the Bands progressed through the layers one by one activating new layers of Neurons as more data was fed into its Channel.

The concept of a "depth of *personality* trait" was adopted for the new Channel Band design. With the "depth of *personality* trait" idea the more a particular hardware was used the stronger that feature became part of the device's *personality.* It can be visualised as a bar chart turned 90°, where the further the bar is to the right-hand-side, the deeper or stronger that trait is in the *personality* of the device (see *Fig.9*).

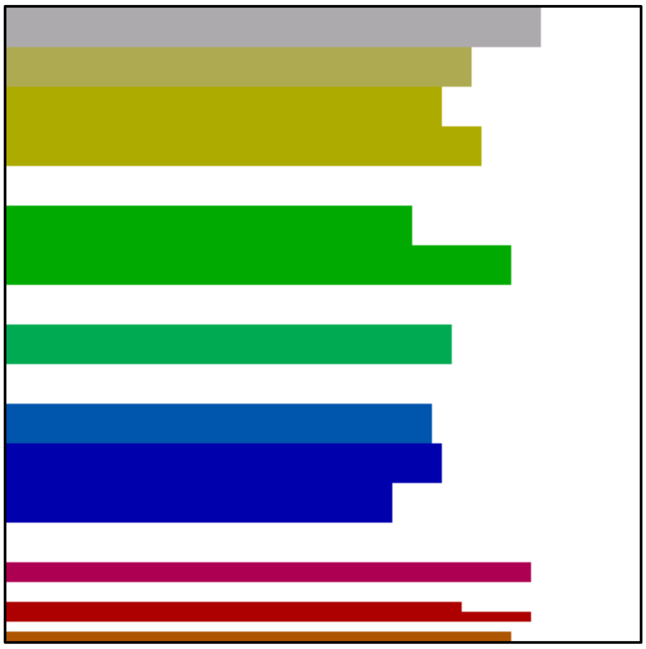


Fig.9. EPID in design phase 3.

These design changes meant that there was less computational overhead due to a reduction in the number of Synapses for Neurons to fire along and less weight values to be stored. Moreover, the new revision meant that a more logical EPID was being returned by the AONN.

The figure (*Fig.9*) represents the Bands' progression from left to right, or from input layer to output layer. Bands nearer the top are thicker than those nearer the bottom. This is due to the fact that there are more Neurons in the Channels' range (see *Fig.3*). This design returned one value for each Channel representing the maximum layer that the band had reached. Thus, the maximum number of combinations or identities possible for this design was 64^{22}. The machine-readable version of the EPID at this stage vaguely resembles an IPv6 address:

```
2:31:0:0:19:60:55:12:41:47:52:0:13:29:39:57:0:0:12:0:0:0
```

Each number separated by a "**:**" represents the current maximum evolution point of a Channel at the point at which the Snapshot is taken.

Channel Bands regress if a layer's time segment elapses with no activity. Depending on the apportioned time segment at a particular layer a proportional decrease occurs in the Channel Band. If the Neurons in a layer are commanded to regress then that Neuron becomes inactive again and can only be activated by receiving input.

In the case of a Band reaching all the way to the output layer, a "cooling off" process is initialised. The cooling off process entails increasing the length of time segment at each layer by two until the Band has regressed to a manageable state.

Third Design Revision

The changes made during the second design revision provided some improvements to the overall resultant EPID. There remained an issue, however, with the amount of information the EPID provided. Thus, the third revision of the AONN design involved adding another dimension of detail to the EPID. The extra detail would ensure that a greater number of unique EPIDs could be generated. With the addition of a variation in tonality within each Band, a higher granularity resulted. Each Neuron's weight value was represented as a variation in the main colour of the Channel's Band (see *Fig.10*).

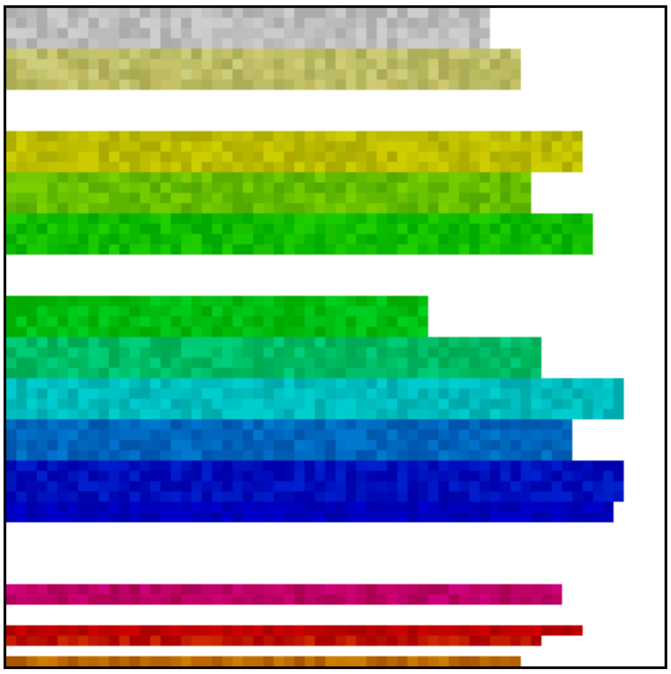


Fig.10. EPID with Band granularity.

This means that the EPID can be used at either of two levels of detail. The short EPID could be used during the initial route discovery phase of a routing protocol. A node could broadcast a search packet to all neighbours, containing its short EPID. Once the node has established which device would be best suited to routing its load it could send its long EPID to the selected device as a means of ensuring it does not select the source node as the next hop. This very basic example would require a Snapshot of the EPID to remain unchanged throughout the duration of the communication.

Final Design Revision

The final revision of the AONN design involved introducing a scaled down version of the AONN with dimensions 32 x 32. This decision was again made based on the consideration of computational expense. Also it is quite probable that an EPID with 8^{1024} possible combinations would provide enough uniqueness to suffice for the purposes it was designed for.

For the scaled down version of the AONN the number of Channels is reduced to 11, however the same Channel Set proportions are maintained. Therefore, the first 6 Channels comprise the first Channel Set with each Channel feeding 4 Input Neural Agents each; the second Channel Set contains three Channels each feeding two Neurons; and the third Channel Set contains 2 single Channels. The overall effect is a less detailed EPID (see *Fig.11*).

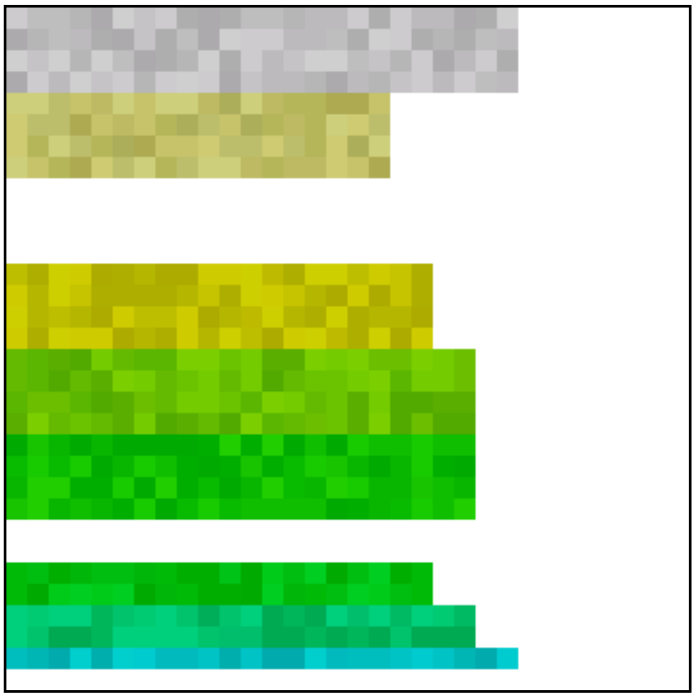


Fig.11. 32 x 32 dimension EPID.

The maximum evolution time of the EPID remained the same at 32 days for this design, which meant that the allocated time segment for each layer was increased proportionately over the smaller Network. A change in the way the Agents were implemented meant that a slightly different approach was taken to the time constraint and progression of a feature's "depth of *personality* trait". *Fig.12* illustrates how a Neural Agent at layer L_n must fire a defined number of times before the following layer, L_{n+1}, can fire. The degree to which the Neural Agent fires is an exponent of the maximum weight held at a Neuron. This way the weight received by a Neuron is normalised and once a threshold value is breached it fires an output weight value of "1" to the next layer. The effect of this is a more steady progression and regression in a Channel Band. The AONN with this approach acts like a signal filter or dampener.

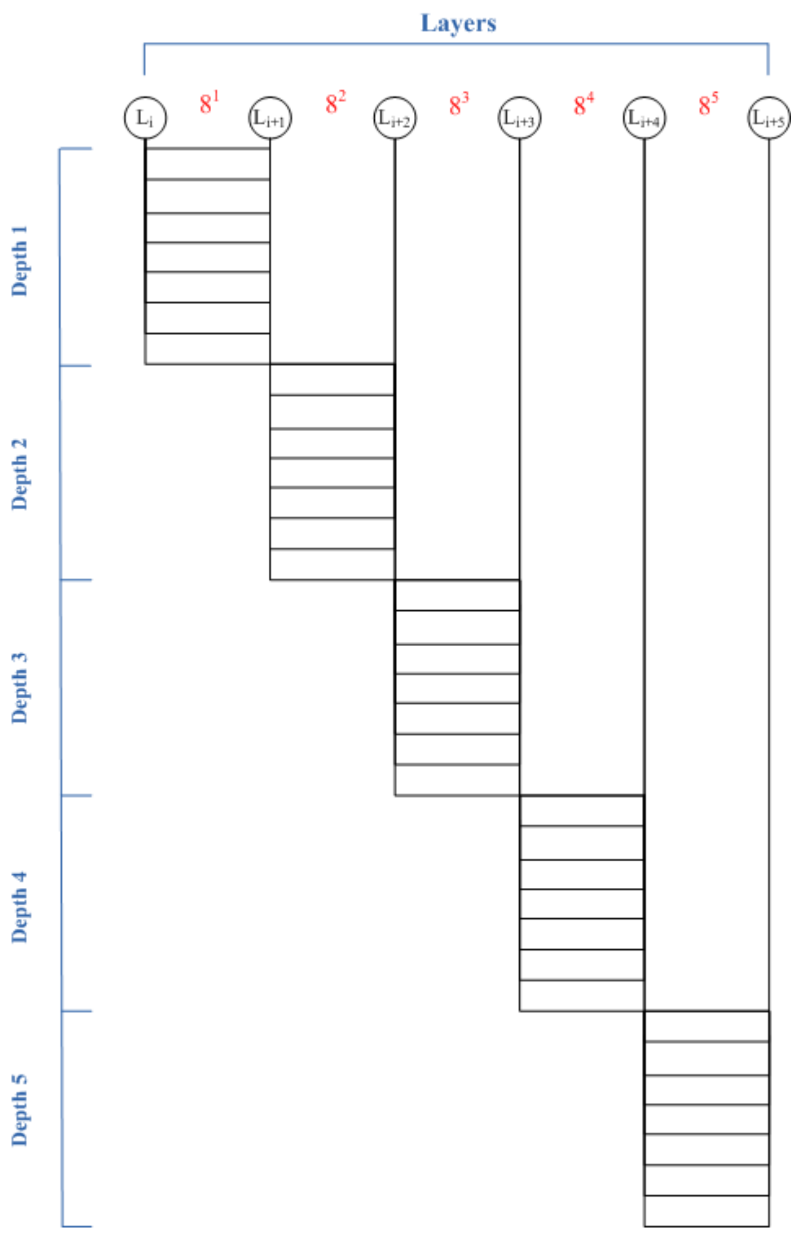


Fig.12. Representation of "depth of personality trait".

The y-axis in *Fig.12* represents the depth of a trait or amount a particular hardware module is used. This is represented by the number of times a Neuron must fire before the following layer fires. The x-axis represents the layers of Neurons.

An equal time segment is apportioned to each layer of the AONN for this design. The flow of Input to the Network is regulated at set intervals. Channel Bands can regress if a set time elapses. With each elapsed time interval the weight value of a Neuron is reduced. When all Neurons within the same range, within the same layer are reduced to 0, the Band regresses by 1 layer. In the event of a Channel Band reaching the output layer the same aforementioned "cooling off" process is invoked. During this design increment the cooling off process consisted of a halving of the time interval to be elapsed in order to incur regression of a Band. The same pattern of Channel Band progression is adhered to in reverse for Channel Band regression. For instance the Neurons at layer L_{n-1}, where n is the maximum number of layers, will only have to display inactivity for half the elapse interval 8^1 times before a full layer is regressed. This regression pattern continues until the Channel Band regresses by a maximum of 25% in a given time frame. This number is selected somewhat arbitrarily for the purpose of demonstrating Channel Band regression as a feature of maintaining an actively evolving EPID. 25% regression equates to 16 layers for the 64 x 64 dimension AONN and 8 layers for the 32 x 32 dimension version. This is considered a reasonable maximum regression as it means that the maximum regression time,

$$T_{\max} = \sum_{i=1}^{\theta(r)} \frac{8^i \cdot t}{2}$$

where, t is the time interval elapsed and $\theta(r)$ is the maximum regression layer. For example, if you consider a device that uses a cellular radio constantly – with the evolution time segment $t_e = 1s$ – a Band will fully evolve in $t_e \cdot 8^{64}$ seconds. Progression and regression are demonstrated in *Fig.13* below.

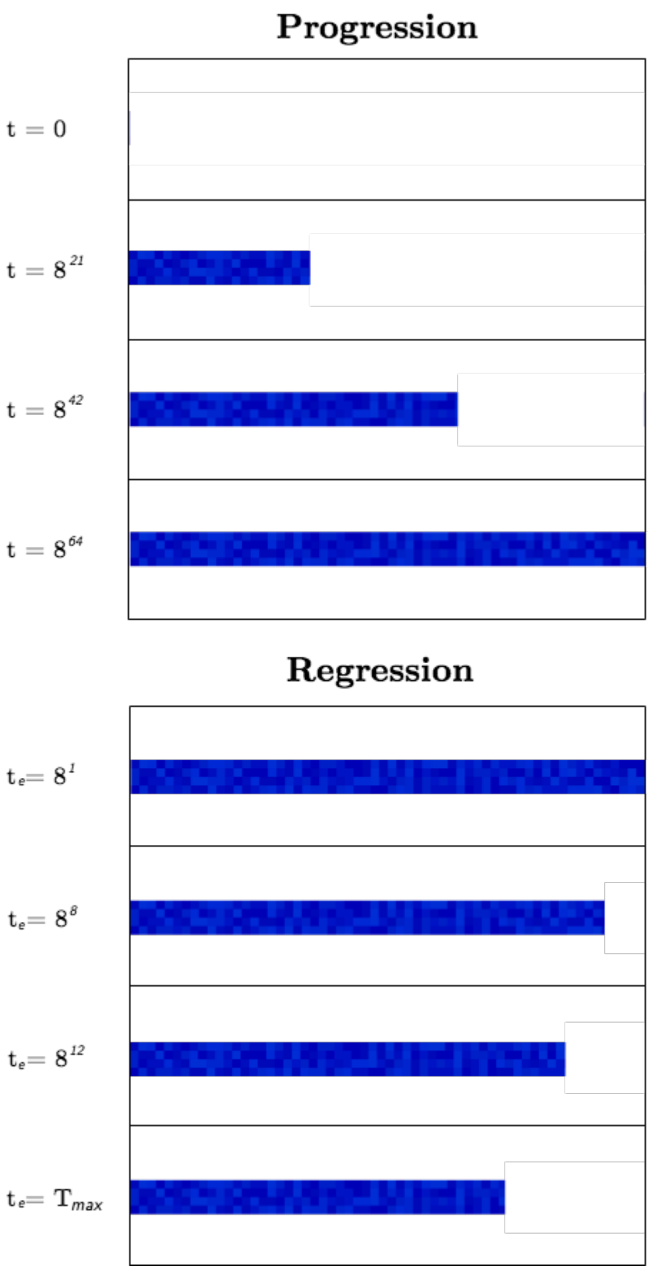


Fig.13. Progression and Regression.

The final state of the AONN is designed such that the EPID is build up from the input layer to the output layer (left to right). When an input

Neural Agent decides to fire it resets its weight value to “0” and continues to process input. Once a Neural Agent fires for the first time it activates the next subsequent layer of Neurons. This increases the depth to which a “trait” features in the *personality* of a device. If a hardware module is not frequently active the Channel Band representing its activity will regress toward the input layer. If a particular hardware device is so active that it reaches the output layer then the Channel undergoes a cooling off period, which stabilises the Channel Band.

Chapter 4

Implementation

In this chapter details about the implementation of the solution are given. Some design choices from chapter 3 are explained further in this chapter.

4.1 Overview

The implementation process for this project followed a rapid prototyping model. For this, software was designed, created and tested in increments. Each design cycle was derived from the previous iteration. Throughout the implementation process there were 8 design-implement-refactor (DIF) cycles. At certain stages during the implementation process a prototype iteration reached a design milestone. These milestones were marked as releases. Each new iteration was an improvement on the previous implementation and with it brought new knowledge, which benefitted each subsequent iteration. The entire implementation process was a learning experience and with each new lesson came new understanding about the problem and new improvements to the solution.

The initial implementation phase was carried out in the Netbeans IDE 6.9 on an Apple MacBoolPro with a 3.06GHz Intel Core 2 Duo processor and 4GB of DDR3 RAM – running OS X 10.6 (Snow Leopard). This implementation phase was designed to provide the proof of concept solution for the AONN and did not take into consideration any processing power restrictions. Java SE SDK 6 was chosen as the development platform due to familiarity with the platform. The AgentFactory Standard Edition (AFSE) framework was chosen for implementing the Agent-based functionality. This decision was justified by two main factors: it is Java based and it also features a java mobility edition.

The Second implementation phase was carried out in the Eclipse Helios 3.6.0 IDE on a HTC Desire Smartphone with Qualcomm QSD 8250 1 GHz (Snapdragon) and 576MB RAM – running Android OS 2.2 (Froyo). The Android platform was chosen because of its power and accessibility. It is uses the Dalvik virtual machine and provides a very powerful development platform. Another attracting factor for using Android 2.2 was its strong emergence into the market place over the past year coupled with the rapid growth of its user base.

This second implementation phase involved porting the proof of concept solution to the Google Android Platform 2.2 and optimizing it for the reduced power hardware.

4.2 Implementation Design

The implementation of the AONN is designed with a number of key components. The components comprise a hierarchical structure, which is reflected in the project's package structure:

- AONN
    - Network
        - Network Utilities
        - Network Elements
            - Network Element Utilities
    - Channels
        - Channels Utilities

These components contain the elements that make up the functioning AONN.

The Network component contains the Network interface, which defines the methods for creating and accessing the structure of the AONN.

The Network Utilities component contains a class, which implements the network interface, called `NetworkUtilities.java`. It is this class, which manages creation of the structure of the AONN. All references to the AONN datastructure are done via the Network interface. This component also manages the EPID Snapshots – both human and computer readable versions.

The Network Elements component contains all of the Network's functional elements:

- Agent interface
- InputNeuralAgent
- HiddenNeuralAgent
- OutputNeuralAgent

The Agent interface defines an Agent object. The InputNeuralAgent, HiddenNeuralAgent and OutputNeuralAgent classes implement the Agent interface.

The Network Element Utilities component contains a class called `Position.java`, which stores an Agents location for context purposes.

The Channels component contains the Channel object class, which is assigned a colour and a number. It is responsible for reading data from a data file and feeding it to the AONN.

The Channel Utilities component contains the ChannelsStandards class and the MonitorUsage class. The ChannelStandards class is responsible for returning the correct range of Neurons for a Channel and also for assigning a colour to a given Channel. The MonitorUsage class is an Android Service, which initializes a thread for each Channel and instantiates each one.

The main elements of the implementation design are depicted in *Fig.14*. The flow of interaction between these elements is also shown.

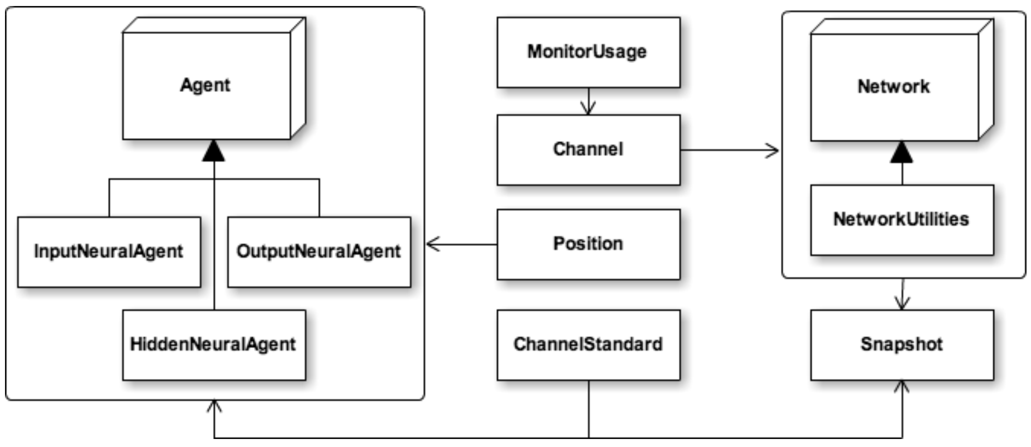


Fig.14. Class interaction diagram.

4.3 Initial Implementation Phase

During the initial implementation phase the goal was to develop a functional proof of concept for the solution. Four DIF cycles were completed during this phase with only one prototype release.

4.3.1 Network Structure

A Network class was implemented to generate and link the structure of the AONN. As the *initial design* incorporated a 64 x 64 dimension EPID, the AONN was thus implemented using a data structure comprised of an ArrayList of 64 ArrayLists each containing 64 Neural Agent objects. A location class was created to store the position of a Neural Agent in the data structure. A Neural Agent object is passed a Position object when it is instantiated. When a Neural Agent fires it invokes the `inputReceived()` method in a Synapse object.

When the `link()` method is invoked in the Network class the synapseLayers data structure is populated with Synapse objects. The data structure contains an ArrayList of 63 ArrayLists each containing 4096 connection objects, which link the Neurons of one layer to those in the next layer. When a Synapse object is instantiated it is passed a source and destination Neural Agent as arguments. When a Neural Agent fires it invokes the `firedVaule()` method in the Synapse object, which passes the weight value received from the source Neuron and passes it to the destination Neuron on the next layer. In accordance with the *initial design* described in Chapter 3, it was necessary to use objects to represent the links between layers because the Synapse links stored the weight values fired by Neurons.

4.3.2 Channels

Channels were implemented as Runnable objects each running in their own separate thread inside a thread pool. 22 Channels were created for the 64 x 64 dimension network and each one bound to a separate hardware module. For the purposes of proof of concept, hardware access was simulated by reading streams data from flat files.

4.3.3 Simulated Data

A method was invoked during the construction of the AONN, which populated 22 flat files with 100,000 binary data instances each. A 1 signifying the hardware module is active and a 0 signifying that it is inactive.

4.3.4 Neural Agents

Neural Agents were implemented with Actuators and Perceptors. The Perceptors define and manage information, which makes up an Agent's Beliefs. Actuators performed the tasks of the Neural Agent (i.e. receive input, fire etc.). Perceptors update the Neural Agents' beliefs according to the Agents' states i.e. fired, not fired etc. This was implemented using a belief statement and by defining a commitment to the task:

```
BELIEF(wantToFire(?name, ?addr)) =>
COMMIT(?self, ?now, BELIEF(true),
    inform(agentID(?name, ?addr), fire)
);
```

Once the Agent has this information then preconditions are defined to control the action of "firing". The preconditions in this case means that the action of firing is not considered until its belief conditions are satisfied.

```
ACTION fire {
    PRECONDITION BELIEF(true);
    POSTCONDITION BELIEF(true);

    CLASS actuator.Fire;
}
```

The Belief is bound to by the Agent by defining its state when it is in an *alive* state:

```
        ONTOLOGY alive {
            PREDICATE wantToFire(?name, ?addr);
        }
```

In the Actuator the definition of the Agents' firing action is defined simply by sending a message to the Agents in the next layer:

```
public boolean act(FOS action) {
      for (int i = 0; i < Network.synapses.size(); i++) {
            Network.synapseLayers.get(position.layer+1)
            .get(i).firedValue(weight);
      }
      return true;
  }
```

When the Neural Agent receives input it checks its beliefs and checks the time since last firing. TemporalBeliefStores facilitate the validation of a belief if a certain temporal constraint is satisfied. If all preconditions and beliefs are fulfilled then the Agent can fire.

4.3.5 Initial Issues & Solutions

With this fully connected neural network implementation there were a total of 262,166 objects being instantiated before any input processing even began in the AONN. With a thread pool of 22 threads continuously feeding data into the Network before long there were multiple `StackOverflow` errors being thrown. Some of the initial `StackOverflow` errors were related to the fact that weight values were not reset or normalised when the firing threshold was reached at a Neuron. The weight values to be stored at Synapses were stored as Integer data types, however when all 4096 Synapses had received a weight value, by the 4th or 5th layer the weight values were getting too large to be stored as Integers. At each subsequent layer a Neuron receives an exponent of weight value stored at a Neuron on the previous layer. For instance, consider an atomic instance of the worst-case scenario, where all 22 Channels feed positive activity data (i.e. a "1") as input to the network, at the same time, for one iteration. If all Input Neurons fired to 4096 Synapse objects each of the Neurons in the next layer receives a weight value of 4096. For the next layer a value of 4096^2 is fed into Neurons on the following layer etc. until by the third layer the weight value being stored is 68,719,476,736. As Java Integers store 32-bit signed numbers the `StackOverflow` error is thrown. The remedy to this problem initially was to normalise the weight and scale them to within the range 0 – 7 to be stored as a whole Integer value at Synapses. Also weight values were reset when they breached the set threshold at each Neuron. These changes provided more stable and consistent weights. Then a colour was assigned to each weight value.

The main cause of `StackOverflow` errors however, was due to the volume of method invocations happening too quickly across too many objects. The number of Neuron and Synapses objects being accessed concurrently by Channels at such a frequent rate was the main cause of the stack running out of memory.

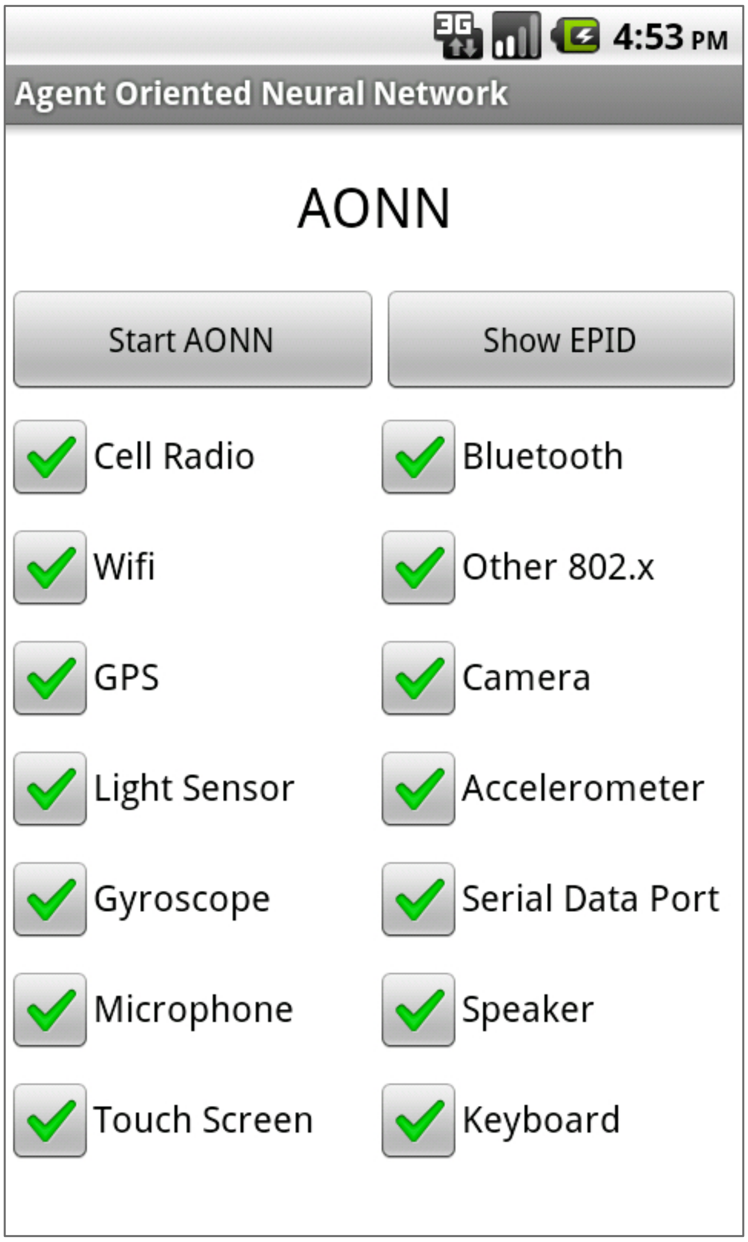
The solution to this issue was to store the weight values at the Neurons rather than at Synapses and reduce the amount of processing required within Synapse objects. Furthermore, the EPID being returned by the AONN was a 64 x 64 matrix of weight values in the range 0 – 7, rather than a 63 x 4096 matrix with the same weight values.

4.4 Second Implementation Phase

During the second phase of implementation the remainder of the development process took place on the Android 2.2 platform. The Android SDK was installed in Eclipse Helios via the Android plug-in. An application was created for Android 2.2 with target API level 8. Three release prototypes were output during this phase with a total of five completed DIF cycles.

The application comprised two Activities: the main Activity and the EPID Snapshot Activity. The main Activity (see *Fig.15*) contains two buttons labelled “Start AONN” and “Show EPID” and also contains a number of check boxes, which allow the monitoring of specific hardware modules to be toggled on and off.

The structure of the AONN is initialized in the `onCreate()` method when the main Activity is built. The “Start AONN” button initializes a Service called MonitorUsage, which instantiates the 22 Runnable Channel objects. Using a Service meant that the hardware monitor could be running even when the Activity it was initialised from did not have focus anymore. By starting threads from the Service it meant that even after the AONN application was closed down it would continue to feed simulated hardware data into the AONN and thus could be running on the device without being affected by the Dalvik garbage collector. As long as the Channel threads were actively accessing the AONN structure, the objects retained their cache allocation.



✓ Cell Radio	✓ Bluetooth
✓ Wifi	✓ Other 802.x
✓ GPS	✓ Camera
✓ Light Sensor	✓ Accelerometer
✓ Gyroscope	✓ Serial Data Port
✓ Microphone	✓ Speaker
✓ Touch Screen	✓ Keyboard

Fig.15. Main Activity.

The EPID Snapshot Activity is executed when the "Start EPID" button is pressed. This Activity renders a humanly readable version of the EPID on an Android Canvas object. Each colour is drawn row by row.

Porting the solution to the Android platform was relatively straightforward from a compatibility perspective. However, immediately there were issues with the cache delegation in the Dalvik virtual machine (VM). As the Dalvik VM is optimised for memory constrained devices the amount of memory assigned to an application is much more restricted. Due to the amount of memory allocation required by the initial design it was necessary to revise the design as describe in chapter 3, beginning with the *First Design Revision.* A number of optimizations were introduced to improve the performance and operation of the AONN for the less powerful platform.

4.4.1 Optimization: Data Type Conversion

The first optimization to be introduced was to replace all Integer data types with Byte data types where possible. All variables whose values were less than 127 were converted to Bytes to save memory. This also impacted on the initial build time of the application because it required less Dalvik cache to be freed before it executed.

This optimization constituted the first DIF cycle.

4.4.2 Optimization: Introduction of Channel Bands

By introducing a system of Bands in the AONN, ranges of Neural Agents were grouped across all layers in the Network. A range of Neural Agents was allocated to each Channel corresponding to the range of InputNeurons each Channel fed. The `getChannelRange()` method in the `com.aonn.channels.util.ChannelStandard` class calculated the correct range for Neurons for a given Channel. The introduction of Channel bands was done for two reasons:

1. To give the EPID a more logical and readable structure and
2. To reduce the number of Synapse objects required per layer.

The latter was imperative for reducing the memory footprint of the AONN. This optimization meant that the number of Synapse objects required was reduced from 4096 per layer to 220. The result of this was that there were now 244,188 less objects being committed to memory before the monitoring and processing of hardware data began.

This optimization constituted the second DIF cycle and first prototype release in this implementation phase.

4.4.3 Optimization: Simulated Agents

The PRISM group at University College Dublin provides a Micro Edition of AgentFactory (AFME), which is based on J2ME. The Neural Agent implementation was altered for the AFME platform. Bridging the J2ME platform with the Android platform is not a straightforward process and would not be recommended. There are some third party bridging APIs available but none of them can port code reliably. An unsuccessful attempt was made to reliably port the code to the Android platform. All that could be achieved was error free code and an attempt to allocate memory to the application, which quit because of excessive requirements. One successful build attempt was achieved on an emulator with all applications deleted and all non-critical processes halted. However, the application crashed when trying to load the view with a `StackOverflow` error being thrown. Further attempts were abandoned, as the requirements for a successful built were unrealistic.

The extra processing cost of utilising the bridge and the computational overhead associated with implementing such a large number of Agents was too great for the test device. As this solution is designed to be viable for devices with restricted memory and CPU power, it was decided to simulate the function of the Agents using the native Android APIs instead.

For the simplicity of the functionality required for the Neural Agents it was decided to simulate the deliberative quality of Agents by implementing Neural Agent objects with similar attributes. This was achieved by ascribing certain criteria to the Neural Agents, which had to be met before the Agent could fire. *Fig.16* shows the structure of the algorithm design to simulate the functionality of the Agents developed in AFSE.

A simplified list of checks was created to simulate an Agent's preconditions. Each of the three categories of Agents was implemented slightly differently depending on whether it was on the input, hidden or output layers.

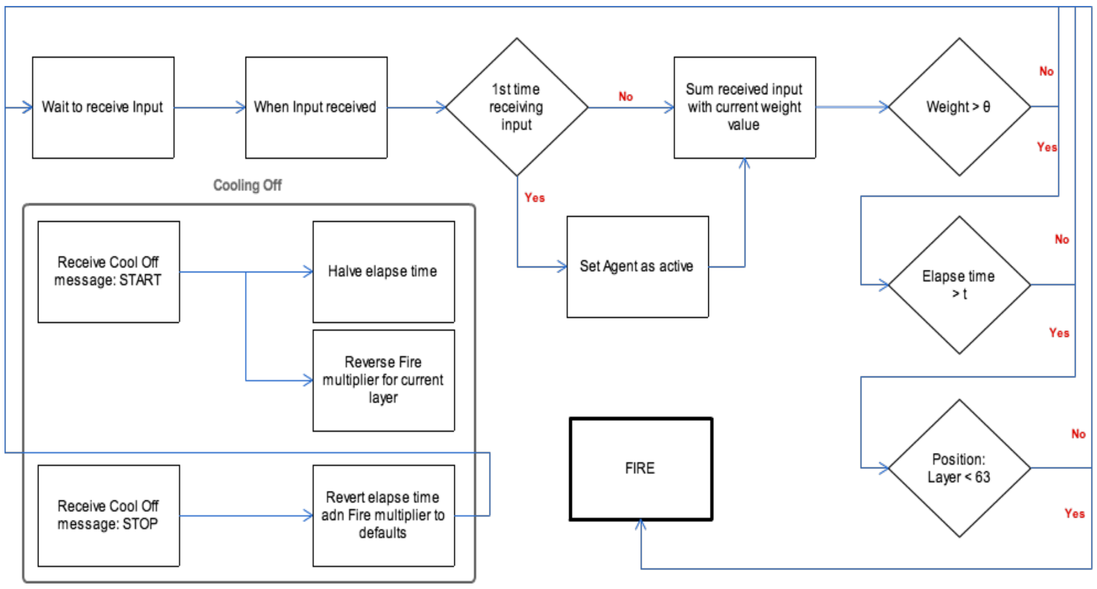

Wait to receive Input
When Input received
1st time receiving input
No
Yes
Sum received input with current weight value
Weight > θ
No
Yes
Cooling Off
Receive Cool Off message: START
Halve elapse time
Reverse Fire multiplier for current layer
Set Agent as active
Elapse time > t
No
Yes
FIRE
Position: Layer < 63
No
Yes
Receive Cool Off message: STOP
Revert elapse time adn Fire multiplier to defaults

Fig.16. Simulated Agent Algorithm structure.

It was found that a 42% decrease in the application's load time was achieved by this optimization. The effect of the desired Agent functionality, provided by the AgentFactory framework, was achieved by the simulated Agents and provided a very large performance increase.

This optimization constituted the third DIF cycle and second prototype release in this implementation phase.

4.4.4 Optimization: Synapse Object Removal

Following the redesign of the Neural Agent implementation, time was given to review the overall structure of the AONN in order to find more opportunities to optimize the solution. As a result of this a decision was made to remove the Synapse objects from the AONN implementation as their function had become trivial. Instead each Neural Agent was programmed to fire directly to the Neural Agents in the next layer, within their assigned Channel Band. This reduced the number of objects being instantiated on initialisation of the AONN by 13,860. What remained at this stage were 4,118 optimized objects comprising the AONN.

This optimization constituted the fourth DIF cycle.

4.4.5 Optimization: Scaled Down Network

When considering that one of the major design goals of the AONN was to provide an inexhaustible address space the 64 x 64 dimension Network certainly provided that. However, with concerns over the performance of the AONN for evolving an EPID a scaled down version of the solution was introduced. A Network dimension of 32 x 32, with 11 Channels was implemented. This further reduced the amount of memory allocation required and reduced the overall running time of the Network. There were still 8^{1024} possible EPID combinations.

In the `com.aonn.Settings` class there is a variable (*NET_DIM*) defined, which can be altered to switch between a 32 x 32 or 64 x 64 dimension network. There is also a variable to adjust the magnification (*MAG*) of the AONN. The recommended values are:

- *MAG = 6* when *NET_DIM = 64.*
- *MAG = 11* when *NET_DIM = 32.*

This optimization constituted the fifth DIF cycle and final prototype release in this implementation phase.

4.5 Final Prototype

The final prototype of the AONN incorporated:

Memory & Storage

- 4096 Neural Agent objects in a nested Collection data structure.
- 22 Channel objects, each run on a separate thread.
- 22 simulated data file.
- Basetime flat file.

Input

- Binary data 1/0.

Output

- Short EPID – either 11 or 22 values separated by ":", signifying the maximum progression of a Channel Band.
- Long EPID – 32 x 32 or 64 x 64 matrix of weight values between 0 - 7.

Operations

- Receive Input.
- Normalise weight.
- Fire.
- Cool off.
- Output EPID.

Chapter 5

Evaluation

This chapter provides a comparative evaluation of the EPID as a means of identifying network nodes. Its suitability as a device identifier for the future of dynamic MANETs with heterogeneous nodes is compared to the current state-of-the-art identification methods.

5.1 Features of AONN

Following the discussion of the AONN's design in Chapter 3 the features of an EPID evolved with an AONN can be defines as follows:

- An EPID is a means of identifying communicating devices.
- An EPID provides a dynamic identity, which can change in different environments.
- The EPID is generated on the device rather than being assigned to it by a network infrastructure node.
- The identity that an AONN returns provides information about the functional capability and use of a device.
- Two levels of EPID detail are available: short EPID and long EPID.
- The AONN can generate 8^{1024} EPIDs for a 32 x 32 dimension AONN or 8^{4096} EPIDs for a 64 x 64 dimension AONN.

The AONN was designed to generate identities for network nodes, which can aid MANET routing protocols by providing them with information about the devices they are routing through.

5.2 Disadvantages

There are a number of disadvantages to the Agent-Oriented Neural Network as a means of generating EPIDs. Also there are some disadvantages to the EPID itself as a universal identification for all types of communicating device.

- The AONN as a mechanism for generating EPIDs remains to be quite resource heavy and may not be suitable for current day reduced performance nodes.
- The standardised assignment of hardware modules to specific Channels in the AONN poses a potential difficulty when considering the shift in consumer trends. Hardware modules that are common in communicating devices today may not be so in 18 months time. Thus, some form of update will be required for assigning new hardware modules to the Channel Standard.
- The long EPID would add a great deal of extra load to the network, as the size of the ID returned by the AONN is very large.
- The time it would take to resolve a node's EPID would certainly be greater than the current time required for other identification method – IPv6 for instance.
- The EPID provides no location specific information. This means that some form of network context information will be required if the solution is to be expanded outside the field of MANETs.

Most of the disadvantages are potential areas of interest for future work but for this research, they remain open questions and distinct disadvantages to the solution in its current state.

When comparing the AONN against other methods of generating node identities, there are some market advantages. Also when comparing the suitability of the EPID as a means of node identification in dynamic

MANETs with heterogeneous nodes, it becomes quite apparent where the advantages lie.

5.3 IPv4 and IPv6

IPv4 is not suitable for use in MANETs because of the static nature of its design. IPv4 addresses define only the physical location of infrastructure nodes such as routers and gateways. No information is provided to the network about the device itself. Similarly IPv6 addresses do not provide information about the device the address is assigned to. There is more flexibility in IPv6 in that routing prefixed can be changed for an entire network without the requirement for renumbering or internal redesign. This means that it is more suitable for dynamic networks. However, there is still the requirement for complex routing protocols for high mobility.

The EPID does not provide any information about the physical location of a node nor does it reflect the infrastructure of the network system it is in. However, in providing information about the type of device being identified and what it is capable of, the EPID offers information that is much more useful for dynamic MANETs with heterogeneous nodes. Devices in a MANETs will be able to identify the capabilities of one another and will be able to make decisions on the most reliable routes to take though a network. This will be possible because selected paths through a network will consist only of nodes with similar EPIDs as the sender. Reliability stems from the assumption that two nodes of equal power and functional capabilities can do similar jobs.

5.4 Mobile IP

MobileIP is an effective way of adapting static IP addressing for added mobility. By introducing a home agent and care-of address a host can move from network to network whilst always maintaining a permanent IP address. A tunnel is used to route datagrams from the home agent to the host's care-of address.

While this protocol is an effective work-around for achieving mobility for IP based devices, there are still issues with smooth hand over between networks and thus fast mobility cannot be accomplished. The added overhead of referring continuously to the home agent and tunnelling datagrams to a care-of address makes it somewhat arduous. However, even with this in place routing within a subnet means nodes rely on regular routing protocols and for this mobile IP does not help.

With the nature of MANETs being autonomous, self-maintained and topology free, routing protocols such as AODV and OSLR help to build route paths using vector distances or link state schemes. These protocols are complex and costly. By providing information about a device to a routing protocol, it would be much easier to decide which next hop to avoid and thus, built of a route path could be reduced greatly.

5.5 Content and Characteristics

Content- and Characteristic-based routing protocols are designed specifically for dynamic MANETs. They take a different approach to routing traffic in topology free networks. Paths are selected based on either the contents of messages being transferred or on some Characteristic such as colours flowing and merging. This research follows the ideology of Characteristic-based routing in so far as it is the belief of the author that a characteristic of a node can aid the routing process. By using a Characteristic of a node rather than a static address routing can be made much easier. However, it is the goal of this research to extend the ideology of Characteristic-based routing to Characteristic-based identification.

For content-based routing a routing decision can be made based on what information is in the data being transmitted. This is interesting because, if the data being transmitted is suitable for one device and not another, the decision to omit that node from the routing path can be made. Much like with the EPID if a node can be identified as being incompatible with the sender then it can be omitted from the routing path.

The overall usefulness of an identification strategy, which can provide useful information about the device being identified, is quite significant. The prospect of being able to select reliable next hop decisions based on the knowledge of a devices' capability affords a great potential for smarter, faster and more reliable routing in dynamic MANETs with heterogeneous nodes. *Table 1* outlines some of the differences between the different identification schemes currently used for routing data.

EPID	**IPv4**	**IPv6**	**Mobile IP**	**Content / Characteristic**
Dynamically evolved on device	Assigned by network host	Assigned by network host	Assigned by home network. Care-of address used away from home	Characteristic of node or content of packet can be used to route traffic
Provides information about device it's on	Provides no information about device	Provides no information about device	Can provide some vendor specific information if using IPv6	Less reliance on fixed state node addressing
EPIDs can be used as a way of preferentially selecting next hop destination	Not suitable for MANETs	Not suitable for MANETs	Cannot facilitate smooth handover or fast mobility	Suitable for MANETs with heterogeneous nodes
Number of possible unique ID will never be exhausted	Address space is already becoming exhausted	Address space is unlikely to become exhausted in foreseeable future	Exhaustion depends on use of IPv4/6	Little reliance on static addressing

Table 1. Comparison between different identification schemes.

Chapter 6

Conclusions

6.1: Conclusions

This research presented a method of identifying network nodes displaying characteristics of its own functional capabilities. A novel concept called an Agent-Oriented Neural Network was introduced. The design of the solution was presented followed by details of its implementation and qualification of some design decisions.

A new identity called an Evolved Personal Identity was introduced, which represented the patterns of hardware usage on a network device. The EPID's advantages and disadvantages were discussed and evaluated against the current state of the art identification methods being used in MANETs.

It can be concluded that the AONN is an effective tool for generating a logical and useful identification for a network node. The processing power required for its operation is perhaps excessive for a large portion of the current network device specifications. However, when more processing power is available to devices this solution could be utilised effectively.

The design of the AONN itself could be improved with some more performance modifications or even by being integrated into the operating system where the hardware monitoring and processing of data could be optimized more aggressively. More care should be taken when planning what technologies should be used for the implementation. Problems with cross platform integration arose when attempting to port the Agent implementation using AFME into the Android platform. A simple fix for these problems would have been to select a mobility platform compatible with the AFME framework such as the Symbian platform.

The resultant EPID from the AONN has shown that it provides meaningful information about the device it is on and is readable both by humans and by machines.

The added load that the EPID would put on a network may hinder its performance and potentially negate its benefits. However, with the option of both a long or short EPID, nodes can avoid incurring the extra cost associated with the long EPID until such time as is absolutely necessary.

6.2: Future Work

The most obvious next step for this research would be to design a routing protocol, which utilises the AONN-generated EPID. An interesting comparison could be made between existing MANET routing protocols and modified versions using EPIDs.

References

Books:

[1] R. Rojas, *Neural Networks: A Systematic Introduction*, Springer-Verlag, 1996.

[2] S. I. Gallant, *Neural network learning and expert systems,* MIT Press, 1993.

[7] S. K. Sarkar, et al., *Ad Hoc Mobile Wireless Networks: Principles, Protocols and Applications*, Auerbach Publications, 2007.

[12] Douglas E. Comer, *Internetworking with TCP/IP - Principles, Protocols and Architecture*, 4th Edition, Pearson Prentice Hall, 2006.

[13] Behrouz A. Forouzan, *Data Communications and Networking*, 4th Edition, McGraw- Hill Inc., 2007.

[14] Murthy C. Siva Ram, Manoj B.S. *Ad Hoc Wireless Networks*, Prentice Hall Communications Engineering and Emerging Technologies Series, 2004.

Papers & Theses:

[3] G. Lei*, et al.*, "Multi-Agent Systems with Local Rules: Towards a Theory of Analysis and Control," in *Control Conference, 2006. CCC 2006.* Chinese, 2006, pp. PL-41-PL-41.

[4] P.F. Driessen, J. Gabert, M.R. Levy, M.H. van Emden, "An Internet protocol for flexible, scalable, and secure interaction with ubiquitous computing devices", *Communications, Computers and Signal Processing, 1997.* '10 Years PACRIM 1987-1997 - Networking the Pacific Rim', 1997.

[5] L. Roberts & T. Merrill, "Toward a Cooperative Network of Time-Shared Computers," *Fall AFIPS Conf.*, Oct. 1966.

[6] L. G. Roberts, "Multiple computer networks and intercomputer communication," *Proceedings of the first ACM symposium on Operating System Principles*, 1967. pp. 3.1-3.6

[8] V. G. Cerf and R. E. Kahn, “A protocol for packet network interconnection,” *IEEE Trans. Comm. Tech.*, vol. COM-22, V5, pp. 627-641, May 1974.

[9] B. M. Leiner, et al., "A brief history of the internet," *SIGCOMM Comput. Commun.* Rev., vol. 39, pp. 22-31, 2009.

[10] J. C. R. Licklider and W. E. Clark, "On-line man-computer communication," *Proceedings of the May 1-3, 1962, spring joint computer conference*, San Francisco, California, 1962.

[15] G. Yang, "ABC: Anonymous routing Based on Characteristics protocol," Masters of Science Dissertation, School of Computer Science and Statistics, Trinity College Dublin, Dublin, 2007.

[16] A. Carzaniga and A. L. Wolf. “Content-based Networking: A New Communication Infrastructure,” In *NSF Workshop on an Infrastructure for Mobile and Wireless Systems*, Scottsdale, USA, Oct. 2001.

[17] R. Chand and P. A. Felber, "A scalable protocol for content-based routing in overlay networks," in *Network Computing and Applications, 2003. NCA 2003. Second IEEE International Symposium*, 2003, pp. 123-130.

[18] A. Carzaniga, A. L. Wolf. "Forwarding in a content-based network," *Proceedings of ACM SIGCOMM 2003.* Karlsruhe, Germany. Aug. 2003, pp. 163-174

[19] A. Carzaniga, M.J. Rutherford, and A.L. Wolf, "A Routing Scheme for Content-Based Networking," *Proceedings of IEEE INFOCOM 2004*, Hong Kong, China. March, 2004.

[20] M. Petrovic*, et al.*, "Content-based routing in mobile ad hoc networks", *Mobile and Ubiquitous Systems: Networking and Services, 2005.* MobiQuitous 2005. The Second Annual International Conference, 2005, pp. 45-55.

[24] Y. Shoham, "Agent-oriented programming," *Artificial Intelligence,* vol. 60, pp. 51-92, 1993.

[25] Rao, Anand, S. and Georgeff, Michael P. "BDI Agents: From Theory to Practice", *Proceedings of the first international conference on Multi-Agent Systems*, ICMAS-95, 1995.

Internet page and Web Articles:

[11] The Internet Engineering Task Force (IETF). Retrieved March 2010: *http://www.ietf.org*

[21] “Gartner Says Worldwide Mobile Device Sales Grew 13.8 Percent in Second Quarter of 2010, But Competition Drove Prices Down”, Press release, Gartner Incorporated, 12 August, 2010. Retrieved August 2010: http://www.gartner.com/it/page.jsp?id=1421013

[22] “Worldwide Converged Mobile Device (Smartphone) Market Grows 56.7% Year Over Year in First Quarter of 2010, Says IDC”, Press Release, The International Data Corporation, 07 May 2010. Retrieved August 2010: http://www.idc.com/getdoc.jsp?containerId=prUS22333410

[23] “Market Information and Statistics (STAT)”, International Telecommunications Union, 2009. Retrieved August 2010: http://www.itu.int/ITU-D/ict/statistics/index.html