2019

Anomaly Detection in BACnet/IP managed Building Automation Systems

Matthew Peacock
Edith Cowan University

Anomaly Detection in BACnet/IP managed Building Automation Systems

by

Matthew Peacock

A thesis presented for the degree of

Doctor of Philosophy

School of Science

Edith Cowan University

Perth, Western Australia

2019

Abstract

Building Automation Systems (BAS) are a collection of devices and software which manage the operation of building services. The BAS market is expected to be a $19.25 billion USD industry by 2023, as a core feature of both the Internet of Things and Smart City technologies. However, securing these systems from cyber security threats is an emerging research area. Since initial deployment, BAS have evolved from isolated standalone networks to heterogeneous, interconnected networks allowing external connectivity through the Internet. The most prominent BAS protocol is BACnet/IP, which is estimated to hold 54.6% of world market share. BACnet/IP security features are often not implemented in BAS deployments, leaving systems unprotected against known network threats. This research investigated methods of detecting anomalous network traffic in BACnet/IP managed BAS in an effort to combat threats posed to these systems.

This research explored the threats facing BACnet/IP devices, through analysis of Internet accessible BACnet devices, vendor-defined device specifications, investigation of the BACnet specification, and known network attacks identified in the surrounding literature. The collected data were used to construct a threat matrix, which was applied to models of BACnet devices to evaluate potential exposure. Further, two potential unknown vulnerabilities were identified and explored using state modelling and device simulation.

A simulation environment and attack framework were constructed to generate both normal and malicious network traffic to explore the application of machine learning algorithms to identify both known and unknown network anomalies. To identify network patterns between the generated normal and malicious network traffic, unsupervised clustering, graph analysis with an unsupervised community detection algorithm, and time series analysis were used. The explored methods identified distinguishable network patterns for frequency-based known network attacks when compared to normal network traffic. However, as stand-alone methods for anomaly detection, these methods were found insufficient. Subsequently, Artificial Neural Networks and Hidden Markov Models were explored and found capable of detecting known network attacks. Further, Hidden Markov Models were also capable of detecting unknown network attacks in the generated datasets.

The classification accuracy of the Hidden Markov Models was evaluated using the Matthews Correlation Coefficient which accounts for imbalanced class sizes and assess both positive and negative classification ability for deriving its metric. The Hidden Markov Models were found capable of repeatedly detecting both known and unknown BACnet/IP attacks with True Positive Rates greater than 0.99 and Matthews Correlation Coefficients greater than 0.8 for five of six evaluated hosts.

This research identified and evaluated a range of methods capable of identifying anomalies in simulated BACnet/IP network traffic. Further, this research found that Hidden Markov Models were accurate at classifying both known and unknown attacks in the evaluated BACnet/IP managed BAS network.

Acknowledgements

I would first like to thank my parents, Anne and Andrew who have always supported me in my endeavours. Their consistent encouragement, love and belief in me helped me through the times where I lacked those qualities. Next, I would like to thank my Grandfather, Ted, who inspired me to pursue science and technology. I miss talking to you every day. Thanks goes to the love and support of my family, who are always there for me and were understanding of my limited time commitments over the duration of my studies.

To my principal supervisor Mike, thank you for guiding me through this research journey, and providing me with so many opportunities over the years. You have influenced how I approach many aspects of life, and I will fondly remember our long discussions on wide and varied topics. My associate supervisors, Craig and Peter must also be acknowledged. Craig, thank you for the many opportunities provided as part of the Security Research Institute, I am forever grateful. Peter, while you joined my panel later into my journey, your support and advice were freely offered long beforehand, and I would like to thank you for that. Thanks are also due to Tony Watson, who provided a timely, thorough proof of my thesis.

I would like to thank ECU and the School of Science for the opportunity to undertake this research degree. Further, I would like to thank the Security Research Institute where I have spent the majority of my time, and made many friends the past few years. My Fellow PhDs at SRI who I have shared the many highs and lows with, particularly Priya, Clinton and Ahmed, thank you for your friendship and advice. I would also like to thank Ghaleb for his assistance in obtaining the real network data for this research, and his helpful discussions in regards to mechanical systems.

Further thanks to Mike, as well as Milan, and Jerry who provided me with the opportunity to be an intern at the Technical University of Eindhoven for 3 months early in my candidature. I learnt so much from the experience, about myself and academia that would not have occurred otherwise. Finally, thanks to the friends I made in Eindhoven, particularly Stefan, Pia, Omer, Mehdi, and Davide, I will always remember the fun we had on Thursday quiz nights at the bar!

Contents

List of Figures

List of Tables

Publications

1. Peacock, M. & Johnstone, M. N. (2014). An analysis of security issues in building automation systems. In *Proceedings of the 12th Australian Information Security Management Conference* (pp. 100–104). Perth, Western Australia. Retrieved from http://ro.ecu.edu.au/ism/170

2. Johnstone, M. N., Peacock, M. & den Hartog, J. (2015). Timing attack detection on BACnet via a machine learning approach. In *Proceedings of the 13th Australian Information Security Management Conference* (pp57–64). Perth, Western Australia

3. Baig, Z. A., Szewczyk, P., Valli, C., Rabadia, P., Hannay, P., Chernyshev, M., ... & Peacock, M. (2017, August). Future challenges for smart cities: Cyber-security and digital forensics. *Digital Investigation*, *22*, 3–13. doi:https://doi.org/10.1016/j.diin.2017.06.015

4. Peacock, M., Johnstone, M. N. & Valli, C. (2017). Security Issues with BACnet Value Handling. In O. Camp, P. Mori & S. Furnell (Eds.), *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - Volume 1: icissp,* (pp. 546–552). INSTICC. SciTePress. doi:10.5220/0006263405460552

5. White, T., Johnstone, M. N. & Peacock, M. (2017, May). An investigation into some security issues in the DDS messaging protocol. In C. Valli (Ed.), *Proceedings of the 15th Australian Information Security Management Conference* (16, pp. 132–139). doi:10.4225/75/5a84fcff95b52

6. Peacock, M., Johnstone, M. N. & Valli, C. (2018). An Exploration of Some Security Issues Within the BACnet Protocol. In P. Mori, S. Furnell & O. Camp (Eds.), *Information Systems Security and Privacy* (Vol. 867, *1*, pp. 252–272). Springer. doi:10.1007/978-3-319-93354-2

Chapter 1

Introduction

1.1 Background

The total cost of reported cyber crime in Australia during 2016 was \$4.3 million USD per year, with an average of two cyber attacks per week per business (Ponemon Institute, 2016). While these are direct costs, indirect or second order costs can manifest when the systems under attack manage cyber-physical devices; such as buildings. Building Automation Systems (BASs), also referred to as Building Management Systems (BMSs), are a collection of devices and software used to automate the control of various services which comprise a building. Automation control was originally used for heating ventilation and air conditioning (HVAC) services, but has steadily evolved to encompass other building services with the improvement of electronic circuitry and microprocessors. BASs are now used to control energy management, water/waste systems, lighting, security and life-safety systems in addition to HVAC, a typical BAS is shown as Figure 1.1. The purpose of a BAS is essentially to optimise the operational costs and processes of a building in relation to the comfort, safety and security of the buildings occupants, goods, and services provided within. Given that 40% of the worlds energy costs are generated from commercial buildings, BAS can provide a significant reduction in operational costs, through the optimisation of building services using sensing devices and efficient load algorithms (IEA, 2015). Further, the BAS market is expected to grow from \$6.65 billion USD in 2016 to \$19.25 billion USD by 2023 (Wood, 2017).

Connections between BASs and other networks, such as enterprise networks and the Internet are increasing. The core motivations for increased connectivity are reduced costs and optimised automation through remote management, outsourced cloud analytic platforms, and the future use in smart cities and the Internet of Things (IoT) (Khaund, 2015; Baig et al., 2017). However, with increased connectivity and automation, network complexity can increase (Kastner, Neugschwandtner, Soucek & Newman, 2005; O'Neill, Bailey, Dong, Shashanka & Luo, 2013), and a need exists for increased device and network level monitoring. While system monitoring is often undertaken for operational means at a device level, network level monitoring is often ignored, which is of principal interest for network security and digital forensic purposes. The lack of network monitoring can be attributed to a historical remnant of the closed system topology, whereby access was physically restricted and thus monitoring for network security was not of concern. When externally connected, there exists a detrimental level of network monitoring in BAS, with Kovach (2016) stating that on average a successful adversary is connected and operating on a BAS without detection for 243 days. Coupled with tens of thousands of misconfigured BAS networks directly connected over the Internet (Peacock & Johnstone, 2014; Johnstone et al., 2015; Praus, Kastner & Palensky, 2016; Gasser et al., 2017), BASs are accessible to adversaries.

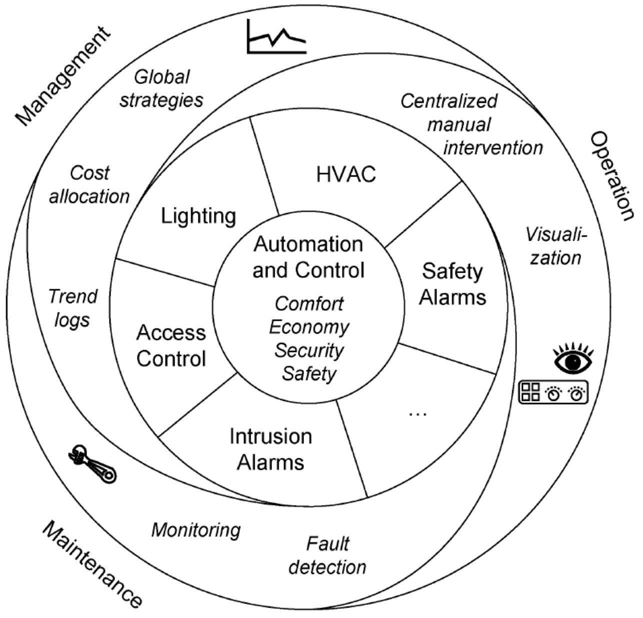


Figure 1.1: The Integrated Building Automation System, replicated from Kastner, Neugschwandtner, Soucek and Newman (2005, p1180)

With increased accessibility comes increased cyber security threat vectors. These vectors are a cumulative effect of the original design of BAS, namely, high trust devices, isolated topology and extensive lifecycle. Previously, adversaries required a physical presence within, or near the BAS for malicious action, which significantly reduced the actionable threats against BASs. With interconnected systems the physical barrier has been removed, exposing BASs to domain specific threats, in addition to inheriting traditionally IT based threats with the adoption of protocols such as IP. BASs face both first and second order threats. As BASs are cyber-physical devices, malicious action against a BAS can cause physical damage to buildings, termed cyber-kinetic attacks (Applegate, 2013). This class of attack can be devastating economically as directly, the damage to the building is sustained, and indirectly, the occupants, goods and services provided by the building are reduced, causing lost opportunity costs. A range of identifiable outcomes for launching a cyber attack against a BAS exist. There are tangible benefits, such as corporate espionage, terrorism or data exfiltration. There are also hidden, second order benefits, such as increasing operational costs and a businesses bottom line, or reputation damage from discomfort caused from the air-conditioning being off in commercial complexes. In both cases, tangible and hidden, there is an associated direct, or underlying cost to cyber attack.

There is an increasing understanding that securing cyber-physical systems is of high importance, highlighted by a number of IoT based Distributed Denial of Service (DDoS) attacks, most prominently the *Mirai Botnet*. The *Mirai Botnet* used a large number of physical devices with low compute power to create a 600Gbps DDoS against a range of services (Herzberg, Bekerman & Zeifman, 2016). It is not inconceivable that a similar event could occur in a BAS, due to the shared design philosophy and characteristics between BAS and other, cyber-physical systems. A high profile example of a BAS attack was conducted in 2014 against the US retail chain Target. The BAS of a Target store was breached using a third party contractors credentials to the HVAC system. The BAS was used as a pivot point into the corporate network, where the point of sale systems in most US Target stores were infected with malware, resulting in 40 million credit card details being stolen, and personal data relating to 70 million customers exposed (Vijayan, 2014; Krebs, 2014). In 2015 after the Target breach, a single credit card detail was reportedly worth between $5 and $30 USD, dependent on the issuing bank, credit card information and geographical location (Mcfarland, Paget & Samani, 2015). Thus, the initial cost of the Target breach was between $200,000,000 and $1,200,000,000 USD. In addition, the second order costs included banks in the U.S spending $200,000,000 USD on bank card replacements, the CEO and CIO of Target losing their positions, and the sales in Target dropping 46% in the quarter after the attack, representing reputation damage. The attack was not detected by Target, but rather Target were notified by the U.S Department of Justice. The success of the attack was due to a combination of lack of network segregation between the BAS, enterprise network and point of sales systems, lack of BAS network monitoring, and the remote access credential theft from a third-party managing the HVAC system.

A focused attack against a BAS occurred in November 2016, when reports emerged of a BAS being attacked in two apartment blocks in Finland (Roberts, 2016). The attack was originally claimed as a DDoS against the building controller, which effectively shutdown the ability to heat the buildings during winter. However, the attack was detected by the BAS management after the BAS "began issuing strange alarms and could not be remotely accessed" Roberts (2016), inferring the adversary had control of the BAS, rather than just DDoSing the BAS. The attack lasted for one day; with resolution achieved via shutting down the BAS and reconfiguring at a hardware level (Roberts, 2016). Envisage this sort of attack against a major apartment block in a global city such as New York, London or Sydney, reputation damage alone would be significant. The response time to the Finland incident was extensive in terms of a control system, whose core purpose is availability. A requirement in any digital system is knowing what your system entails, and having oversight to ensure operation is running accordingly. The Finland example reveals a core issue in a security context for network oversight in a BAS. The attack was detected due to the out-of-bounds nature of the communications being sent, which was identified in the normal monitoring of the system for performance and general operation. Without actively monitoring for security, a *stuxnet* type attack, where normal inbounds commands are acting maliciously could be envisaged to exist on a BAS.

Table 1.1: Survey results, adapted from Facilities.net (2015)

Question	Yes	No	Not Sure	N/A	Total Respondents
Are any of the building automation systems in your buildings connected to the Internet?	84%	16%	-	-	224
If the building automation system(s) is (are) on a dedicated building automation network, is it bridged to the corporate/enterprise network?	35%	29%	27%	9%	173
Has a budget been established for security countermeasures for building automation systems	41%	59%	-	-	172
Have you conducted a threat assessment of your network and physical security measures for cyberattacks on your building automation systems?	42%	58%	-	-	157
Has your building automation system monitored for cyberattacks?	54%	46%	-	-	155
Have you developed a plan for responding in the event of a cyberattack on your building automation system?	37%	63%	-	-	156

While there are known cyber security issues, awareness in the BAS domain has been increasing. A 2015 survey conducted by *Facilities.net* recorded 224 building operation managers as respondents on cyber security and BAS trends, selected results are detailed in Table 1.1. Of note, 75% of respondents stated their organisation did not have a formal cyber security incident response plan for their BAS. 66% of respondents were not confident in their organisations ability to effectively recover from an attack, and 84% of respondents stated their BAS was connected to the Internet.

An understanding of the protocols and components operating in the BAS domain is required to detect attacks against the system. A shift to open-source protocols over the past 30 years has changed the landscape from primarily vendor-specific proprietary protocols, to three major open-source protocols and a range of proprietary protocols which encompass the market, namely, BACnet, KNX and LONWorks. The Building Services Research and Information Association (BSRIA) states that BACnet is the dominant communications protocol with a 54.6% world market share as of 2015 (Towler, 2015). Following this trend, BACnet is widely deployed in government, industry and businesses around Australia. As such, the research presented focuses on the BACnet/IP protocol, which uses IP as its communication media.

The SANS 2016 state of ICS security survey, which includes BAS, states that 54% of respondents are reliant on their trained staff to search and detect threats manually, while 30% state they use anomaly detection tools (Harp & Gregory-Brown, 2016). Given the time taken to detect attacks on BAS, there exists the need to improve detection methods beyond manual means, and account for the potential of *stuxnet* type zero day attacks.

Classification of the ability to detect threats faced by BASs can be represented using a Johari window (Luft & Ingham, 1955). Kim (2017) adapts the Johari window model to present certainty and identification in terms of risk, shown as Table 1.2. Kim (2017) elaborates on "unknown unknowns", through a further classification of unknown-unknown types, namely, unidentified due to knowledge gap, or unidentified due to an assumption. These include time space or condition, between parts or the whole system. Identifying hidden or temporal interactions in a system can

Table 1.2: Johari window of certainty and knowledge, replicated from Kim (2017, p155)

		Certainty	
		Known	Unknown
Identification	Identified (Known)	Known-Known (Identified Knowledge)	Known-Unknown (Identified Risk)
	Unidentified (Unknown)	Unknown-Known (Untapped Knowledge)	Unknown-Unknown (Unidentified Risk)

be achieved through learning how a system operates in relation to its defined rules and semantic meaning. Methods which can account for temporal features, such as time series analysis, state space modelling and some machine learning algorithms could be appropriate.

Intrusion Detection Systems (IDS) for computer networks have been an active research field since the initial works of Anderson (1980) and Denning (1987). The purpose of an IDS is to identify potential attempts of unauthorised access or actions being undertaken on a system (Denning, 1987; Yu & Tsai, 2011). Typically, classification of IDS approaches fall into two categories, misuse-based (often called Signature) and anomaly-based (Axelsson, 2000). Misuse-based rely on accurate signatures of malicious actions, to which normal traffic is compared. Comparatively, anomaly-based approaches define a baseline of normal actions on the network through a learning phase, and report deviations from normality as potential intrusions.

Application of Intrusion Detection Systems to BACnet managed BAS is a growing area in cyber security literature, with many authors focusing on anomaly detection (Kaur, Tonejc, Wendzel & Meier, 2015; Tonejc, Güttes, Kobekova & Kaur, 2016; Caselli, Zambon, Amann, Sommer & Kargl, 2016; Esquivel-Vargas, Caselli & Peter, 2017). The application of machine learning to enhance intrusion detection is a more recent occurrence. Tonejc et al. (2016) explores unsupervised machine learning methods to identify anomalous traffic in BACnet/IP networks. Application of supervised learning, which can account for temporal data has not been explored.

1.2 Purpose

"There are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns – the ones we don't know we don't know." Rumsfield (2002)

This research identifies a gap in knowledge in regards to anomaly detection in BACnet/IP managed Building Automation Systems (BASs). The literature review reveals limited applications of intrusion detection methods against BACnet/IP managed BASs. A similar problem faced by generic IDS researchers is suffered in BAS anomaly detection, the lack of datasets. Datasets in BAS anomaly detection can be classed as a real network dataset, requiring synthetic malicious traffic, and small-scale simulated network datasets with limited attack variance. More recent studies utilise both types of dataset to improve the efficacy of the presented detection method (Tonejc et al., 2016; Esquivel-Vargas et al., 2017). Further, the application of machine learning use in IDS for BAS is limited, with investigations focused on unsupervised methods. Results from the literature show the ability to detect *"known known's"* with varying success rates, but rarely address *"unknown unknowns"*.

This study aimed to investigate methods of identifying both known and unknown BACnet/IP specific attacks in a BACnetwork. Given the temporal nature of BAS, models which can utilise time based features, such as Markov Models, time series analysis and Artificial Neural Networks were explored and compared. As with previous studies, a range of real and simulated datasets were obtained to evaluate the methods. The purpose of the research is thus to improve anomaly detection in BACnet/IP managed building automation systems to improve the protection of critical infrastructure. A range of research questions were derived to further define the research, detailed in §1.3

1.3 Research questions

RQ_1 *How can known and unknown attacks against BACnet/IP based Building Automation Systems be detected?*

- SQ_1 *Are BACnet devices exposed to known threats?*
- SQ_2 *Do known BACnet attacks have distinguishable network patterns compared to normal BACnetwork traffic?*
- SQ_3 *Is machine learning applicable to identify known and unknown attacks against BACnet/IP networks?*
- SQ_4 *How accurate are machine learning approaches in detecting known and unknown attacks against BACnet/IP networks and devices?*

1.4 Thesis terminology

While the writing convention of this thesis is Australian English, many BACnet components are referred to throughout the thesis in American English spelling in line with the SSPC-135 (2012) standard. Further, BACnet networks will be referred to as BACnetworks.

1.5 Thesis structure

The remainder of the thesis is structured into a number of distinct parts. Chapter 2 reviews the literature in the domain of Building Automation Systems and approaches undertaken in Building Automation cyber security. Further, approaches for state modelling and machine learning applied to cyber security problems, and the applications to building automation systems security is discussed. Chapter 3 details the research design and methodology used throughout the research. A discussion of the dominant research paradigms is undertaken, culminating with the paradigm selected for this research. Further, a review of approaches resolves the specific methods used throughout the research, forming the research design. Chapter 4 outlines the exploratory results of the research, including a survey of Internet connected devices, threat modelling of devices and simulation design. Chapter 5 continues, presenting the results of a range of algorithms applied to the generated simulation data. Chapter 6 discusses the results in respect to the existing literature, followed by a critical review of the research process. Finally, Chapter 7 presents conclusions drawn from the research, and suggests potential future work in the domain.

Chapter 2

Literature Review

The literature review chapter frames the research project undertaken in the context of previous works in the respective areas. The chapter begins with a discussion of Critical Infrastructure (CI), Supervisory Control and Data Acquisition (SCADA) systems and where Building Automation Systems (BASs) sit in relation to CI and SCADA. Followed by a discussion of the open-source protocol landscape; accompanied by a comparison and justification of research towards BACnet. The review then traverses to BACnet security, outlining the current state of the BACnet protocol in regards to security features. Following, a critical review of previous and current works in BACnet security, with particular focus on intrusion detection research is presented. Machine learning techniques for anomaly detection, and system behaviour are discussed, with specific applications applied to BACnet identified. Finally, three machine learning algorithms are presented as potential approaches to identify anomalies in BACnet/IP managed BAS.

2.1 Critical Infrastructure

Critical Infrastructure (CI) is defined by the Australian Federal, State and Territory governments as

> *"...those physical facilities, supply chains, information technologies and communication networks which, if destroyed, degraded or rendered unavailable for an extended period, would significantly impact on the social or economic wellbeing of the nation or affect Australia's ability to conduct national defence and ensure national security"* Attorney Generals Office (2010, p. 8).

The direct mention of "physical facilities" should be evidence enough that buildings are a core part of CI. However, it was not until 2015 that in the United States, The National Institute of Standards and Technology (NIST) updated the "Guide to Industrial Control Systems (ICS) Security", to mention Building Automation Systems (BASs) as an "other type" of control system classed as CI (Stouffer, Pillitteri, Lightman, Abrams & Hahn, 2015). Further, the SANS 2016 State of ICS Security Survey encompassed BAS as a core focus for the survey, alongside SCADA, Distributed

Control Systems (DCS) and Process Control Systems (PCS) (Harp & Gregory-Brown, 2016). The classification, and inclusion as a core focus underpins the importance of securing Building Automation and Control systems, which have been used extensively for over 30 years. There has been a gradual convergence between Information Technology (IT) and Operational Technology (OT) (Murray, Johnstone & Valli, 2017). The convergence between building controllers and IT based protocols has created a larger threat surface, not previously faced by building controllers. Differing control structures however have left traditional cyber security approaches in the IT sector incapable of direct application securing building controllers. These issues faced are similar in nature, and general operation to Supervisory, Control and Data Acquisition (SCADA), systems; another CI, which has been the focus of security researchers for some time.

2.1.1 Supervisory Control and Data Acquisition

SCADA systems are used to monitor and control industrial automation processes which are geographically dispersed. In Australia, the distance between stations can be thousands of kilometres. SCADA systems are deployed to manage utilities and CI, including gas, water, electricity and traffic systems. SCADA systems provide a real-time centralised monitoring and control system for large numbers of process inputs and outputs (Stouffer, Falco & Kent, 2007). SCADA systems were originally implemented with proprietary protocols for communication; however over the past two decades, standardised network communications using TCP/IP has increased (Zhu, Joseph & Sastry, 2011). Standardisation to IP protocols and connecting SCADA systems to the Internet for remote management and control can leave SCADA systems exposed to vulnerabilities. The barrier to entry for attackers is reduced due to external facing connections, rather than needing line of sight or physical access to systems. Traditional IT security measures, such as patching directly clash with the high availability requirement of SCADA, further exacerbating security concerns with patchable vulnerabilities existent on these systems (Dussel et al., 2010; Murray et al., 2017).

2.1.2 SCADA security

Due to the cyber-physical nature of SCADA systems, cyber security is of high importance (Zhu et al., 2011). Previously, threats to availability were a hardware reliability issue, with protection tied to fault prevention and detection (Cárdenas, Amin & Sastry, 2008). However, with increased network connections in SCADA systems and increased use of commercial off the shelf (CoTS) IT systems, protection against cyber threats is now also required to prevent availability issues (Cárdenas et al., 2008; Dussel et al., 2010). Unlike CoTS devices, the application of CoTS based cyber security practices are limited by the SCADA environment. Noted by Granzer and Kastner (2010), limitations include power requirements, security scalability, communication medium support, the use of non-IP protocols and quality of service differences. The potential impact on availability that security practices create is often the limiting factor upon direct integration of security systems to SCADA. Patching often requires system downtime which is not acceptable in life safety or CI systems particularly (Zhu et al., 2011; Murray et al., 2017). The culmination of these issues has resulted in a high number of legacy devices in SCADA systems, which are unpatched (Cárdenas et al., 2008; Cheminod, Durante & Valenzano, 2013; Murray et al., 2017). In addition, protocol design can contribute to system vulnerabilities in SCADA. For example, ModBus, a widely used SCADA protocol contains no confidentiality, integrity or authentication checking processes (Benbenishti, 2017), allowing attackers to use the normal commands of the protocol to undertake malicious actions. Building Automation suffers from many of the same security issues as SCADA systems (Fisk, 2012). Particularly protocol based vulnerabilities, caused by initial design choices stemming from a segregated local network with low access and highly trusting devices.

2.2 Building Automation Systems

Building Automation Systems (BASs), historically called Building Management Systems, are the result of the centralisation of control and management of services operating in a building. Originally, BASs encompassed heating, ventilation and air conditioning (HVAC) systems, but now can be used to centralise control of lighting, water systems, power management and security features, such as CCTV cameras and access control (Kastner et al., 2005; H. Merz, 2009). Typically, a BAS consists of management devices, controllers and field devices, such as sensors and actuators. The services in buildings operate using schedules and control loops, defined for the specific facility based on the features of the building, and surrounding environment. The primary goal of BASs is to reduce the cost of a building, through the application of smarter control loops and schedules to reduce energy consumption, while maintaining comfort for building occupants (H. Merz, 2009). For this purpose, data collection is required from field level devices to make short term decisions in controllers, and long term decisions based on trend data. Trend data is compared to the control schema, when the trend does not follow the control schema, it is a point of interest to investigate, showing either the system is incorrectly programmed, or an extreme weather event caused the controls to activate out of bounds. However, there is the potential that trend data could also indicate a network attack event against a device. Typically, the network data associated with a BAS is not monitored, with the interest from owners and operators of the system coming from data values, rather than network traffic (Caselli, 2015; Jabado, 2017; Humphries, 2017). Correlation between network traffic and trend data could be used to identify issues from a security perspective.

Emerging from a proprietary locked industry, three open source network protocols with similar aims have become dominant in BASs, namely, BACnet, KNX and LONWorks. The largest protocol is BACnet, with a world market share of 54.6% in 2015 (Towler, 2015), thus BACnet is the object of interest for this research, due to its proliferation in Australia and abroad. Byres, Franz and Miller (2004) note that SCADA protocol adoption and use is highly coupled to industry preference, vendor operating requirements and design history of systems. These attributes are applicable to BAS protocols, which followed a similar history of proprietary communication protocols in closed internal networks.

There are a number of challenges facing BASs, particularly the IT/OT convergence also faced by SCADA systems. The long life-cycle of buildings, compared to IT devices is of note. Typically, BASs are expected to operate for 10-20 years, this time frame requires designs to be flexible for future advancement and integration, while also supporting legacy devices and maintaining a limited amount of interaction for security purposes (Kastner et al., 2005) BAS design is thus in a challenging position, given the advancement of the Internet of Things (IoT), Smart Cities, Cloud based analytics and Wearable technologies, which can be used to improve BASs core purpose (Baig et al., 2017; Ahmad, Mourshed, Mundow, Sisinni & Rezgui, 2016). From a security standpoint, the incorporation of these devices, which were designed to be interconnected, into the BAS domain which was typically locationally isolated increases the threats faced exponentially. Fisk (2012) noted that external connection is required for systems to maintain longevity to allow for future patching cycles. Counter to this, Cárdenas et al. (2008) state that "Patching and frequent updates, are not well suited for control systems" (Cárdenas et al., 2008, p3), citing a nuclear power plant which accidentally shut down after a diagnostic computer rebooted for a software update. Given the range of vendors, devices and protocols in use, every BAS is a unique implementation. Each system now requires knowledge and management of IT domain systems and issues, while maintaining a coexistent relationship with the safety and operational constraints of an OT system.

2.2.1 System requirements

Apart from the functionality of controlling building services, BAS also have a number of requirements with regards to safety and operation. The requirements from OT do not always overlap with IT and more specifically, security. Novak and Treytl (2008) discuss the common requirements between safety and security systems, summarised in Table 2.1. Novak, Treytl and Palensky (2007) describe the comparative goals between safety and security, which can allow for crossover life-cycles to be developed, as undertaken in Novak and Treytl (2008) and Novak and Gerstinger (2010), with further discussion in Cheminod et al. (2013). Novak and Treytl (2008) elaborate that safety and security systems have a common goal in reducing risk; but state that confidentiality and non-repudiation are not relevant for automation systems. While confidentiality may be an added burden to low-power systems, the data transmitted is important and should be protected. Additionally, non-repudiation of commands sent on a BAS is an important feature which would be required for a secure system. For a safety system, it would be required to know with great certainty that a device had acted according to its specification and cannot deny its involvement in an action. Further, Cheminod et al. (2013) highlight the key disparity between requirements in BAS and IT are the real world ramifications of system failure. Downtime in IT systems is generally acceptable, and often to be expected in relation to the patching cycles of software. Exemplified by service level agreements for Internet service providers stating a 99.999% uptime on their service (Amazon, 2017). Additionally if an IT device fails, the device is generally survivable when proper contingency planning was implemented. However, if safety or security critical systems fail, the risks are significant, with the potential for loss of life, asset damage and environmental impact. These issues cause BAS to follow the paradigm that failure is unacceptable, similar to SCADA systems

(Kirsch, Goose, Amir, Wei & Skare, 2014). The difficulty arises when implementing this paradigm using failure tolerant IT systems.

Table 2.1: Summary of security and safety common requirements defined in Novak and Treytl (2008, p312)

Requirements of	**Requirements of Safety Systems**			
Security Systems	Integrity	Authentication	Availability	Authorisation
Confidentiality				
Integrity	✓			
Availability			✓	
Authentication		✓		
Authorisation				✓
Non-Repudiation				

2.2.2 Protocols

BAS specific protocols are typically open source, examples include BACnet, KNX, LONWorks and ModBus. In addition, a number of proprietary frameworks interlinking the open source protocols together exist, such as Tridium-Fox and Schneider's Continuuum. The benefit of the three major BAS specific open source protocols, apart from being device independent, is the ability to communicate with multiple protocols, on existing cabling infrastructure (Granzer, Kastner & Reinisch, 2008), reducing costs significantly. While originally serial-based, other data media such as Ethernet are increasingly being used for BAS, with IP encapsulation and more recently native IP stacks allowing BAS to be connected to enterprise networks and allow direct remote access over the Internet (Kastner et al., 2005; ASHRAE, 2018).

2.2.2.1 BACnet

Building Automation Control Networking (BACnet) is an object-oriented peer-based protocol for managing BASs, which began development in 1987. The aim of BACnet was to create a protocol which would work with management, field and automation devices, and provide interoperability between other protocols. BACnet was released in 1995 as an ASHRAE/ANSI standard, in 2003, BACnet gained ISO standardisation (ISO 16 484-5). BACnet is actively maintained, with reviews of the protocol occurring every four years until 2008, thence changing to biennial reviews (SSPC-135, 2017). The most recent BACnet standard is BACnet-2012, Version 1, Revision 19 (SSPC-135, 2017). In December 2016, Addendum *135-2016bj* was released for initial public advisory, as of June 2018, Addendum *135-2016bj* is in its second round of public advisory. To provide interoperability, BACnet focusses on the Network layer and above, allowing for multiple physical and data link layer technologies to be used (Hersent, Boswarthick & Elloumi, 2012). While there are defined protocols in the standard for interoperability depicted in Table 2.2, further undocumented mapping can occur due to the flexibility of the protocol (Granzer et al., 2008; Hersent et al., 2012). BACnet utilises UDP for transmission over IP networks, using a virtual IP network stack. UDP is used to reduce the overhead which occurs from using connection orientated protocols, such as TCP (Zachary, Brooks & Thompson, 2002), along with providing the ability to map protocols to interpret the UDP data (Newman, 2013). However, Addendum *135-2016bj* aims to implement a native IP option as opposed to Virtual IP, to allow for greater interconnection between traditionally BAS devices, and IT services such as cloud analytic platforms.

Table 2.2: BACnet architecture mapping to the OSI model as of revision 19, replicated from SSPC-135 (2012, p11)

OSI	**BACnet Layers**						
Application	BACnet Application Layer (APDU) Application						
Network	BACnet Network Layer (APDU) Network						
Data Link	ISO 8802-2		MS/TP	PTP	BVLC	LonTalk	ZigBee Data Link
Physical	Ethernet	ARCNET	EIA-485	EIA-232	UDP/IP		802.15.4 Physical

2.2.2.2 KNX

Konnex (KNX) Association began in 1999, as a merger between the European Installation Bus (EIB), Batibus and European Home System (EHS) protocols. The aim of KNX was to define and offer certification services for the KNX open standard (Hersent et al., 2012); which was defined in 2002 (Granzer et al., 2008). KNX became a European Standard in 2004, and defined as ISO/IEC 14543-3 in 2006 (ISO, 2006). KNX follows the OSI model for packet construction, see Table 2.3. Further, KNX is based on the older protocol EIB; allowing EIB to coexist and be compatible with KNX. Similar to BACnet, KNX is flexible in physical media usage, with twisted-pair, power line and wireless radio available. Additionally, KNX can use tunnelling to operate over IP, with unicast for configuration and maintenance, and multicast for process data exchange (Granzer et al., 2008). KNX has a small network stack, making it suitable for field devices with low processing power, while also providing collision avoidance via CSMA/CA (Granzer et al., 2008).

Table 2.3: KNX/EIB architecture mapping to the OSI model, adapted from Köhler (2008, p12)

OSI	**KNX/EIB Layers**					
Application	KNX/EIB Application Layer					
Transport	KNX/EIB Transport Layer					
	Connection Orientated			Connectionless Orientated		
Network	KNX/EIB Network Layer					
Data Link	MAC through CSMA/CA					
Physical	Twisted Pair		Power Line		Radio Frequency	Ethernet
	TP-0	TP-1	110Khz	132Khz	868Mhz	UDP/IP

2.2.2.3 LONWorks

Local Operating Networks (LONWorks) is a network platform developed by Echelon Corp., and was accepted as an ANSI standard for control networking in 1999 (Hersent et al., 2012). In 2008, LONworks was approved as ISO standard ISO/IEC 14 908-1, -2, -3 and -4 (Hersent et al., 2012). LONWorks aim was to move away from proprietary centralised control models, by using connection devices to exchange data directly, eliminating the need for a central controller, and thus the single point of failure (Hersent et al., 2012). LONWorks is the combination of the LONTalk communication standard, defined as ISO/IEC 14908 ('ISO/IEC 14908 Information Technology - Control network protocol', 2012) with Neuron chips developed by Echelon Corp. Originally dominant in the transportation and utilities industries, the LONWorks platform was adapted to BASs (Snoonian, 2003). Similar to BACnet and KNX, LONWorks is physical media independent, allowing the use of twisted pair, power lines, wireless and optical fibre (Hersent et al., 2012). LONworks is not an IP native protocol, and makes use of ANSI/CEA-852 IP tunnelling to connect LONwork networks to IP networks (Hersent et al., 2012). A mapping of LONworks to the OSI model is shown as Table 2.4

Table 2.4: LONWorks architecture mapping to OSI layers, adapted from Hersent, Boswarthick and Elloumi (2012, p. 63)

OSI	**LONWorks Layers**								
Application	LONWorks Application Layer								
Presentation	LONWorks Presentation Layer								
Transport	LONWorks Transport Layer				OR	LONworks Session Layer			
Network	LONWorks Network Layer								
Data Link	MAC through CSMA/p-persistent								
Physical	Twisted Pair			Power Line				Fibre Optic	LON-Works over IP
	Free Topo-logy	RS-485	Transformer Isolated	75kHz	86kHz	115kHz	132kHz		

2.2.2.4 Protocol comparisons

The objective of all three major open-source protocols is identical, to increase interoperability over multiple physical media and vendor devices, and thus reduce the cost of implementation while also providing robust building automation operation. A shortcoming in all three protocols is that they are based on BASs initial design, being a local, trusted network. With the increased use of the Internet, remote administration and more recently cloud-based analytics and smart city initiatives, BASs have become interconnected to enterprise networks and the Internet (Baig et al., 2017). Security, was not an original function of BASs, nor needed given the previous closed network topology (Newman, 2013). However, with increased connectivity exists an increased attack surface, where a lack of inherent security processes expose BASs to adversaries (Khaund, 2015). Security through obscurity no longer exists in BASs (Kastner et al., 2005), the cyber-physical nature of BASs, and the integration of security and safety services makes the system a potential target to a range of adversaries (Khaund, 2015). Overall, the cyber security practices in BASs are lacking. A range of factors contribute, including but not limited to, a lack of awareness and a limited consolidated effort for implementing network security features.

2.3 BACnet structure

This research focuses on ASHRAE Standard 135-2012 BACnet, as this was the latest version implemented upon undertaking the research. Additionally, the BACnet 2012 is the version to which BACnet compliant devices are measured (BTL, 2017). BACnet is an object-oriented protocol in which associated sets of values are represented by an object, a complete listing is shown as Table 2.5. These objects are held in collections representing a single device, referred to in the object model as a BACnet Device. There are 54 standard objects that communicate using 38 standard services as of the 2012 version of the BACnet Standard (SSPC-135, 2012). When transported on UDP/IP as default, BACnet uses UDP Port 47808 for communications to the server running the service. However, it should be noted that the UDP port in use is arbitrary, with proprietary middleware implementations of BACnet communicating on additional UDP ports (Schneider Electric, 2015; Gasser et al., 2017).

Each object contains a set of properties to describe the object. For each of these properties a definition of type is included, the type could be analogue, binary, text or a number of other possible types. Additionally, each attribute includes a flag indicating optionality, essentially if the property can be omitted and still operate in accordance with the protocol standard. The majority of properties present in each object is optional. To facilitate object handling and communication, each object contains a mandatory property of an object identifier field, which identifies what type of object is being examined. Every BACnet device requires a mandatory device object, which provides addressing details for communication on the BACnet network (SSPC-135, 2012). All other objects are implemented based on the requirement of the device (SSPC-135, 2012). Each BACnet device supports a number of services. These services provide the means of communication between devices on the network. The services make use of a client-server model for connectivity similar to that in use on TCP/IP networks, however the classification is reversed. The clients are controller devices, while servers are sensors, actuators and point controllers which generate the data to provide to the client devices. Thus, server devices have much lower hardware capabilities compared to client devices. As stated, there are 38 services defined in the 2012 version of the protocol, outlined in Table 2.6. The BACnet services themselves are not dependent on any particular network infrastructure, with the protocol specification stating that BACnet is to be agnostic to the transport of data (SSPC-135, 2012). Addendums to the protocol have formally added virtual IP and ZigBee to the list of supported underlying protocols, however in theory any underlying protocol could be utilised for this purpose (Granzer et al., 2008; Hersent et al., 2012).

A minimum set of services required for operation is defined as a BACnet Interoperability Building Block (BIBB) in the BACnet standard. Each BIBB represents a service function, or network

Table 2.5: BACnet standard objects sorted by type, adapted from SSPC-135 (2012), bolded objects represent the most common objects used defined by Distech (2010, p23)

Basic Device	**SimpleValue**
Device	Bit String Value
Analog Input	Character String Value
Analog Output	Date Pattern Value
Analog Value	Date Value
Binary Input	Date Time Pattern Value
Binary Output	Date Time Value
Binary Value	Integer Value
Multi-state Input	Large Analog Value
Multi-state Output	Octet String Value
Multi-state Value	Positive Integer Value
File	Time Pattern Value
	Time Value
Physical Access Control	**Notification**
Access Credential	**Event Enrolment**
Access Door	**Notification Class**
Access Point	Notification Forwarder
Access Rights	Alert Enrolment
Access User	
Access Zone	
Credential Data Input	
Presentation	**Process**
Group	**Averaging**
Global Group	**Loop**
Structured-View	**Program**
Life safety and security	**Logging**
Life Safety Point	Event Log
Life Safety Zone	**Trend Log**
Network Security	Trend Log Multiple
Schedule	**Control**
Calendar	**Command**
Schedule	Load Control
Meter	**Lighting control**
Accumulator	Channel
Pulse Converter	Lighting Output

message which can occur in the device, specifying which device is to act as a client or server for each individual service. Thus there are two types of BIBB, Client BIBBs which represent supervisory/control devices requesting data and actions to occur (classed A), and Server BIBBs representing sensors which respond to requests with data (classed B). The BACnet standard defines six device profiles which couple a set of BIBBs with the minimum working set of objects with which to classify a device. See Addendum A for a complete listing of device profiles. Each device may implement

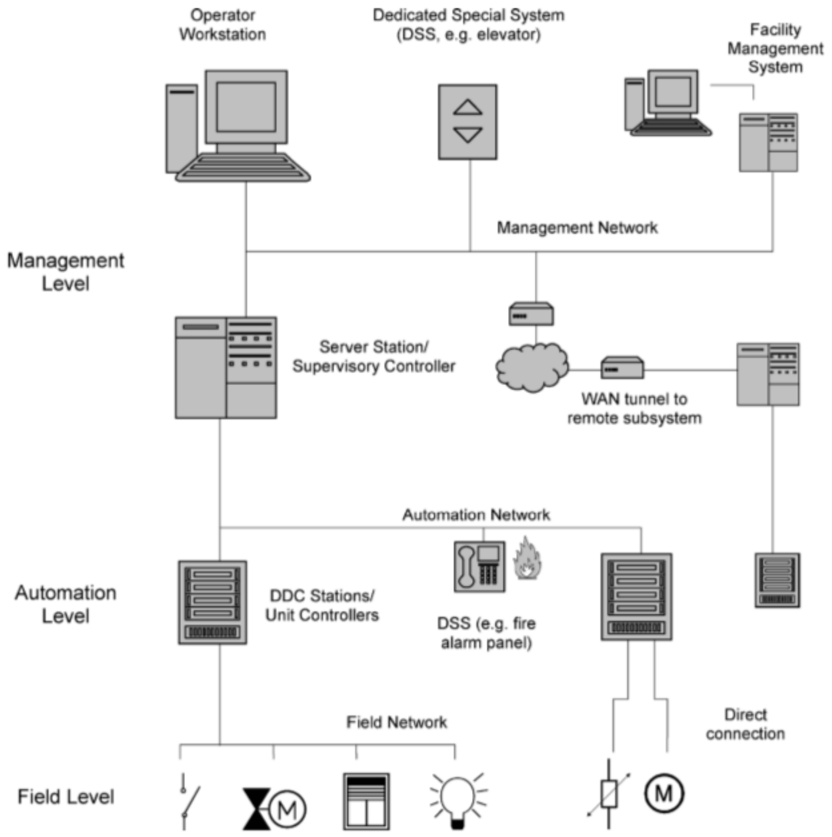
Table 2.6: BACnet standard services sorted by type, adapted from ANSI/ASHRAE standard 135, BACnet 2012. (SSPC-135, 2012)

Alarm and Event	**Remote Device Access**	**Object Access**
AcknowledgeAlarm	DeviceCommunicationControl	AddListElement
ConfirmedCOVNotification	DeviceCommunicationControl	RemoveListElement
UnconfirmedCOVNotification	ConfirmedPrivateTransfer	CreateObject
ConfirmedEventNotification	UnconfirmedPrivateTransfer	DeleteObject
UnconfirmedEventNotification	ReinitializeDevice	ReadProperty
GetAlarmSummary	ConfirmedTextMessage	ReadPropertyMultiple
GetEnrollmentSummary	UnconfirmedTextMessage	ReadRange
GetEventInformation	TimeSynchronization	WriteProperty
LifeSafetyOperation	UTCTimeSynchronization	WritePropertyMultiple
SubscribeCOV	Who-Has	WriteGroup
SubscribeCOVProperty	I-Have	
	Who-Is	
	I-Am	
File access	**Virtual Terminal**	
AtomicReadFile	VT-Open	
AtomicWriteFile	VT-Close	
	VT-Data	

additional services other than the minimum required in the standard, thus comparable devices from different vendors can have differing functionality. Thus, each device of the same profile type may have different objects and services but undertake the same functions, due to a minimum operating specification and optionality of most objects, object properties, and services which can vary by vendor. Compliance with the BACnet standard is based on a minimum set of features required, with devices currently tested against the ASHRAE 135-2012 BACnet version (BTL, 2017). Compliance holds for one specific hardware version of the device, with future versions requiring a new round of compliance testing. To determine the objects and services a specific device supports, BACnet employs a Protocol Implementation Conformance Statement (PICS), which each vendor is required to generate for each device a vendor produces. Each device is therefore capable of generating a PICS, from which core information can be retrieved. Compliance holds for multiple software versions for a device, according to the listing of PICS retrieved from BTL (2018). Thus many devices for sale use older BACnet protocol revisions for operation. Typically, the BACnet protocol version a device operates with is between five and ten years old. The flexibility of the protocol implementation has allowed BACnet to become widely used and interoperable, over a range of devices and use cases, but also makes it difficult to baseline devices for security purposes (Caselli et al., 2016).

2.3.1 Typical network topology

Historically, BACnet operated on a three tiered network topology (Kastner et al., 2005), see Figure 2.1. With the advancement of microprocessors, the typical three tier topology has been reduced



Operator Workstation
Dedicated Special System (DSS, e.g. elevator)
Facility Management System
Management Network
Management Level
Server Station/ Supervisory Controller
WAN tunnel to remote subsystem
Automation Network
Automation Level
DDC Stations/ Unit Controllers
DSS (e.g. fire alarm panel)
Field Network
Direct connection
Field Level
M
M

Figure 2.1: BACnet three tiered network topology, replicated from Kastner, Neugschwandtner, Soucek and Newman (2005, p1183)

to two tiers (Kastner et al., 2005), management and field, with the automation tasks undertaken on point controllers in the field tier and network controllers operating in the management tier, see Figure 2.2. The management tier contains networking controllers, operator workstations, human machine interfaces and facility management networks, which provides the ability to retrieve data and interact with field tier devices to make high level decisions. For interconnected BACnet systems, the management level also holds connections to the enterprise network, and externally to the Internet through a firewall. The field tier represents physical devices in the network which generate data about the environment, or interact with humans, such as sensors, actuators, valves, light switches. The field tier also holds point controllers, such as variable air valve controllers, thermostats and zone controllers, which control subsections of a network allowing for local control decisions (Kastner et al., 2005). Serial connections between field devices are often daisy chained together operating with master/slave token passing, with the network controller acting as a bridge between the serial and digital management network (Cisco, 2008). There has been a shift to BACnet using exclusively IP networks, culminating with the public review release of BACnet Addendum *135-2016bj* (ASHRAE, 2018). Currently, BACnetworks operate over IP using a virtual IP stack, which encapsulates idioms of the BACnet application layers to transport over IP medium. BACnet/IP networks do not use token passing, but rather operate as a traditional peer based network. As such,

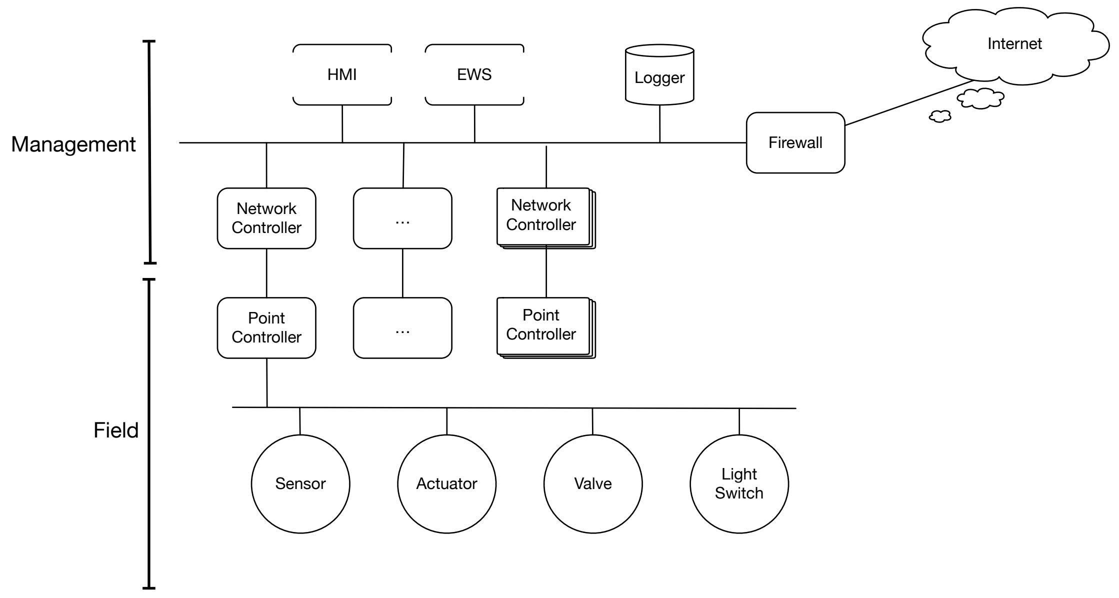


Figure 2.2: Example of BACnet two tiered network topology, described in Kastner, Neugschwandtner, Soucek and Newman (2005)

BACnet/IP networks follow a more traditional IP based network topology, where related devices are in a joint subnet. Typically, this only consists of controllers and management devices such as workstations due to their higher feature set.

2.3.2 BACnet networking devices

In contrast to the conventional definition of a router (separation of networks) a BACnet router translates between two different protocols in a network. When using BACnet/IP, connections to other protocols in the hierarchy, such as BACnet MS/TP or ModBus are connected using a BACnet router (Thomas, 2008). BACnet routers are not necessarily standalone devices, and are often incorporated as part of a BACnet device, typically the network controller.

Given the reliance on broadcasting, a BACnet Broadcast Management Device (BBMD) can be used as a gateway device for BACnet devices that are on different sub-networks. The purpose of a BBMD is to broadcast, direct or relay packets between subnetworks, as BACnet routers simply translate messages between protocols (Thomas, 2008; Distech, 2010). In the same way that Ethernet bridges store Media Access Control (MAC) addresses, the BBMD stores locally known or registered devices in their BACnet Device Table (BDT). Further, devices which operate under a separate networking controller in the shared backbone network are classified as "Foreign Devices", and are stored in a Foreign Device Table (FDT) in the BBMD. BBMD management traffic is undertaken using UDP/IP.

The BACnet standard defines 19 Network layer messages, which are used for device management and network security processes when implemented. In addition, vendors may implement proprietary messages, outlined in Table 2.7.

Table 2.7: BACnet network layer messages defined by SSPC-135 (2012)

Hex Octet	Message
X00	Who-Is-Router-To-Network
X01	I-Am-Router-To-Network
X02	I-Could-Be-Router-To-Network
X03	Reject-Message-To-Network
X04	Router-Busy-To-Network
X05	Router-Available-To-Network
X06	Initialize-Routing-Table
X07	Initialize-Routing-Table-Ack
X08	Establish-Connection-To-Network
X09	Disconnect-Connection-To-Network
X0A	Challenge-Request
X0B	Security-Payload
X0C	Security-Response
X0D	Request-Key-Update
X0E	Update-Key-Set
X0F	Update-Distribution-Key
X10	Request-Master-Key
X11	Set-Master-Key
X12	What-Is-Network-Number
X13	Network-Number-Is
X14 - X7F	ASHRAE Reserved
X80 - XFF	Vendor Proprietary

2.4 BACnet security issues

" We didn't really think that the threat would come from someone tapping into the network in some dark mechanical room and sending legitimate, but malicious, messages." Newman (2013, p. 43)

BACnet was not designed with security as a primary requirement, as the original intention and implementation of BAS was isolated from external connection. With the advancement of networking technology, the change from serial-based networks to Ethernet, and the rise of the IoT, BAS networks now have external facing connections to internal enterprise networks, and the Internet for remote management and cloud-based data analytics (Ahmad et al., 2016; Baig et al., 2017; IBM, 2017). As such, the attack surface against BAS networks, including BACnet managed networks has increased.

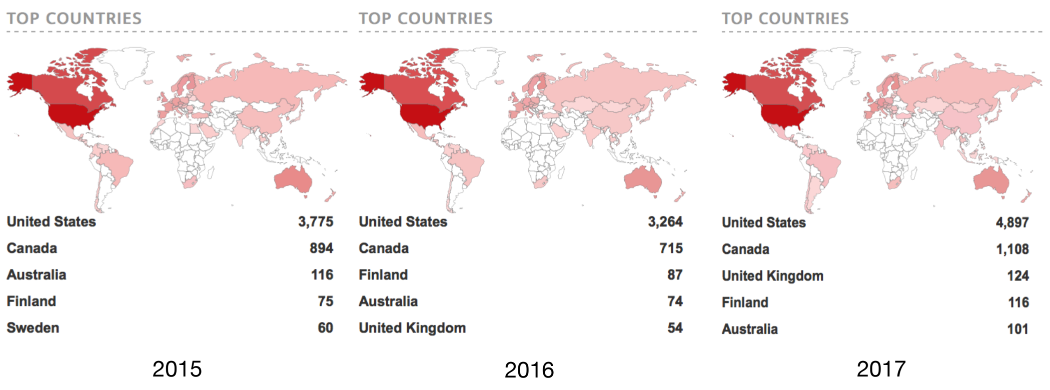
Before publishing the first edition of the BACnet standard in 1995, public review comments highlighted a concern about security in BAS (Newman, 2013). As noted by Newman (2013), peoples view of the "main threat" to BACnet was different. The main threat proposed by the BACnet working group was that of a disgruntled employee, who would use an existing operator workstation to attack the BACnetwork. The realisation of the threat provided by public comments however was an external physical intruder who would tap into the network and send "legitimate, but malicious messages" (Newman, 2013, p43). With the way BACnet is structured, and the now ubiquitous connection to the Internet, the physical requirement of the intruder to send legitimate commands which cause malicious actions has been eliminated. While BACnet has the most robust security features of the three major open source protocols, the lack of implementation equates the robustness to naught.

With the increased connectivity of BAS to other networks, such as enterprise and the Internet, BAS are exposed to the same threats faced by traditional IT based networks and protocols However, BAS are also exposed to their own set of threats, due to a hierarchal topology, broadcast based communications and extensive trust placed in communications from devices connected to the network. The consequence of this design leaves BAS networks with no third party verification for source authentication, exposing BAS protocols to message interception, replay and message insertion attacks (Holmberg, 2003; Granzer, Praus & Kastner, 2010). In addition, BAS are vulnerable to a range of denial of service attacks through normal operation of BAS protocols against specific building services (Antonini, Barenghi, Pelosi & Zonouz, 2014; Mundt & Wickboldt, 2016).

Many researchers have identified and classified vulnerabilities in the BACnet protocol. Holmberg (2003) in their 2003 threat assessment identified a range of vulnerabilities against BACnet, with classification split into two distinct parts, IT based, which represent generic Internet Protocol based vulnerabilities, and BACnet protocol specific vulnerabilities. Within the BACnet vulnerability class, there are five categories, snooping, application service attack, network layer attack, network layer Denial of Service and application layer Denial of Service. Similarly, Kaur et al. (2015) identified three classes of vulnerability against BACnet, adapted from IT (equivalent to Holmberg's IT based), non-conformance and protocol vulnerability. Further, Caselli (2016) defined snooping, Denial of Service and process control subversion. At the time of writing, no research was found to have been undertaken to consolidate a classification scheme, or address the impact of each BACnet vulnerability against a system. The range of vulnerabilities defined have had limited known usage in real networks. Reported attacks against BACnet specifically are limited, and those that exist are not at a level where specific vulnerabilities are discussed. Typically, reported attacks are disclosed by researchers, rather than reported by advisories after an attack has occurred.

Researchers have implemented a range of attacks against BACnet using the detailed specific vulnerabilities in a number of simulated environments (Johnstone et al., 2015; Tonejc et al., 2016; Esquivel-Vargas et al., 2017). Bowers (2013) outlined a range of attacks implemented as part of an automated attack framework implemented in *Python*. The Framework contained three categories, Discovery, Enumeration and Fuzzing, which utilised legitimate BACnet commands. However, often researchers forgo directly implementing an attack, rather using synthetic data manipulation for out-of-bounds data set generation and testing purposes (Kaur et al., 2015; Tonejc et al., 2016).

There are various reports of the number of BACnet devices which are directly accessible to the Internet. Praus et al. (2016) details results from 2014, where 13,964 BACnet devices are openly accessible. More recently Gasser et al. (2017) undertook active Internet-Wide traffic measurements searching for BACnet devices, between 2016 and 2017, revealing 16,485 internet accessible devices. Further, Gasser et al. (2017) scanned 16 ports, as opposed to only the default port 47808, which they identify as 53% of their responses. Identifying an exposed BACnet device is relatively simple given the protocol has high trust, and is verbose. In addition, fingerprinted device search engines such as Shodan, can be used to search directly for BACnet devices. The devices detailed from a Shodan scan is not a complete listing of all devices, given the results displayed are a subset of total results due to caching. At any one point around 3,000 readily accessible BACnet devices exist through the Shodan Engine, with Australia being consistently in the top five exposed countries (Peacock & Johnstone, 2014; Peacock et al., 2017). Figure 2.3 details three Shodan searches undertaken at various points during the research duration. In 2017, Positive Technologies undertook data collation of scans from Shodan, CenSys and Google to outline the number of Internet accessible ICS components. 175,632 devices were found, with BACnet devices accounting for 13,717, and the Tridium Fox framework accounting for 39,168, placed 4th and 2nd respectively out of all protocols detailed (PositiveTechnologies, 2018). With an increased level of exposure, detection of incidents is of importance to reduce associated costs and risks.



United States	3,775
Canada	894
Australia	116
Finland	75
Sweden	60

United States	3,264
Canada	715
Finland	87
Australia	74
United Kingdom	54

United States	4,897
Canada	1,108
United Kingdom	124
Finland	116
Australia	101

Figure 2.3: Shodan search engine results outlining directly accessible Internet connected BACnet/IP devices at the time of query in 2015, 2016, and 2017. The darker the colour the more devices identified.

Second order threats are those which have an emergent effect against a system. In BAS, these can include damage to physical goods/data inside a building, such as perishables or data servers through temperature manipulation. Since BAS networks are connected to enterprise networks, attacks such as data theft and extortion on enterprise networks can also occur via the BAS network (Cárdenas et al., 2008; Vijayan, 2014).

Direct and second order threats are tightly coupled. One vulnerability could exploit each of these threats, for example an attack against a fan in a HVAC, which uses legitimate commands sent from a trusted by default device on the network would impact both the physical device (i.e burn out the fan) but also impact the goods inside the building (Johnstone et al., 2015).

Compared to other connected devices, such as desktop computers and smart phones, BAS devices undertake relatively simple computing tasks, and as such have reduced compute power by design. With the added long life-cycles of BAS networks compared to IT devices, most IT devices have an order of magnitude more processing power and memory size than a building controller. The connection of BASs to external networks provides a means for contextually powerful devices to interact with BAS devices, which through normal interaction can overwhelm BAS devices and cause physical damage to building components and the surrounding environment (Holmberg, 2003; Johnstone et al., 2015).

2.4.1 Summary of core issues

Many of the underlying issues facing BACnet controlled BAS originate from historical design choices based on the then locally connected topology. BACnet devices are peers, meaning there is a high degree of trust placed in the network communications originating from a device. Due to previously being closed topology, source authentication was not required and thus is lacking from base protocol implementations of BACnet. Given there is no third-party source authentication, any device on the network which utilised the BACnet protocol can communicate with other BACnet devices. BACnet has limited knowledge of network scope and segregation, thus if an external device can interact with a BACnet device, it will act on its command. As BASs often run over office network architecture, and provides remote access to internal systems, securing the BAS is important for enterprise network security (Cárdenas et al., 2008; Fisk, 2012), in addition to the security of the BAS. More recently, BASs using BACnet have been integrated into cloud analytic platforms and the IoT (GO-IoT, 2018). Discussion has also begun on the use of integrating wearable technologies to further optimise control procedures in BAS for future smart cities (Baig et al., 2017), opening another avenue into the network which must be secured. Additionally, as BASs are more commonly being connected to enterprise networks, the possibility of pivoting into the secured internal enterprise network can exist (Cárdenas et al., 2008).

Oversight on the network level is also an issue. Typically, BACnetworks have sensor monitoring and data aggregation via trending for operational purposes, but not network level oversight. Generally, the owners and operators of the BAS do not have control over the network, with IT personnel controlling the network (Jabado, 2017; Humphries, 2017). Often BAS operators have a network to operate the BAS, but outsource monitoring of the network connection to the IT department, who do not deploy network monitoring to a segregated part of the network, allowing for malicious network transactions to occur undetected.

2.5 Approaches to securing BACnet

> *"no company has yet implemented it[BACnet security services] in a commercially available product"* Newman (2013, p. 44)

The 2003 threat assessment undertaken by Holmberg (2003) formed the basis of the construction of an improved security addendum for BACnet. Prior to this, Clause 24 of the BACnet Standard outlined minor network security features discussed in Zachary et al. (2002) and Holmberg (2003) as in-secure. The BACnet Security Services (BSS) is the improved security addendum, defined as Clause 24 of the BACnet standard 2012, with the intent to *"...provide peer entity, data origin, and operator authentication, as well as data confidentiality and integrity"* SSPC-135 (2012, p720). Clause 24 specifically mentions it is not defined to provide *"... authorization policies, access control lists and non-repudiation"* SSPC-135 (2012, p720). Security features are implemented as a set of network layer messages, but is typically described as a separate stack layer. Clause 24 provides the ability for devices to authenticate, data hiding and user authentication through the use of shared keys used to sign messages, of which there are six types outlined below (SSPC-135, 2012).

1. General-Network-Access
2. User-Authenticated
3. Application-Specific
4. Installation
5. Distribution
6. Device-Master

The General Network Access Key is provided to all devices, and is used to sign Broadcast network layer messages, enable encryption tunnels and used by user interface devices which cannot authenticate normally (SSPC-135, 2012). The standard notes that messages sent with a general network access key should not have their user ID and user role fields trusted (SSPC-135, 2012). The User Authenticated Key are provided to trusted client and server devices which implement identity management, allowing for trust of the user ID and user role fields. Application-Specific keys provide security boundaries between specific building services, such as HVAC and lighting (SSPC-135, 2012). An Application-Specific Key is thus only provided to devices sharing a building service, with Clause 24 stating these keys can be for highly secure communication, allowing restriction of services initiated by devices with lower privileged keys (SSPC-135, 2012). Installation keys are temporary keys which are aimed to be used by technicians to configure controllers using tools that would not normally access the network (SSPC-135, 2012). Distribution keys are used to distribute all keys bar the device-master over the network, as per local security policy (SSPC-135, 2012). The device master key is used for the distribution key, and can be either a unique prefixed key, or requested from a key distribution server using the Set-Master-Key service (SSPC-135, 2012). Key distribution is intended to occur via a BACnet key server (SSPC-135, 2012). All keys are bundled into a set and distributed with a single key revision number, each device receives a specific set of keys appropriate for the device use (SSPC-135, 2012).

The BSS defines SHA256 or MD5 for key Hashed Message Authentication Codes (HMAC), however, given that MD5 is widely identified as an insecure hashing technique (Turner & Chen, 2011), the integrity of signing messages using BSS is questionable.

Table 2.8: BACnet security policies, adapted from SSPC-135 (2017)

BACnet Security Policies	
Plain-Non-Trusted	Not physically nor digitally secure
Plain-Trusted	Physical security is required, without protocol security
Signed-Trusted	Physical security is not required, digital security is provided by signatures
Encrypted-Trusted	Physical security is not required, digital security is provided by encryption

Encryption is often not implemented on control networks due to the processing capability of devices (Cheminod et al., 2013), and to provide the ability to maintain oversight for safety features. Additionally, encryption does not prevent legitimate messages causing malicious actions if a trusted device is taken over by an adversary. Clause 24 defines four network policies for devices, defined in Table 2.8.

The caveat of BACnet security is that BSS defined in Clause 24 is optional. Noted by Newman the original BACnet working groups director, *"no company has yet implemented it[bacnet secure services] in a commercially available product"* Newman (2013, p. 44). Additional limitations are identified in Clause 24, of note, for secure communications Clause 24 states that not only the signature, but also the *Device ID* is required. Thus, the secure features of the protocol are reliant on knowledge of the *Device ID* field of other devices, which is generally requested before communication via a broadcast. Additionally, advice given in Clause 24 is to disable a range of error conditions to prevent potential denial of service attacks, the equivalent of disabling ICMP in IP networks for diagnostics, which is generally a discredited idea (Scarfone & Hoffman, 2009). To implement Clause 24, the standard defines a minimum device requirement, consisting of the details listed below.

1. Have an application layer
2. Support execution of *WriteProperty*
3. Ability to track time
4. Have non-volatile re-writable storage
5. Not be an MS/TP Slave device

These limitations discount legacy, and current field devices specifically, which generally cannot track time and are MS/TP slaves. From a security standpoint, the development of BACnet seems to have a reduced focus since the inclusion of the BSS as Clause 24 in 2009. Little additional security analysis was performed via the BACnet working groups until 2017, with an updated release of Addendum *135-2016bj* discussing security. As noted, there is a push for BACnet to be part of the IoT. Given the lack of implemented security features, the introduction of BACnet into the IoT area is concerning, and requires further investigation. Currently, the security features of the protocol are assumed to be handled by the IP suite (Newman, 2013). From a security standpoint, there are many questionable functions in the protocol base, which seem fundamentally at odds with traditional IT based security theory and practice, yet are required for BACnet to operate in accordance with the standard; for example, the reliance on broadcast communications (Newman, 2010). The BACnet protocol is intended to provide a base level of functionality, with the flexibility to implement security features on a per-use basis, given the optional criterion of the BSS. The implementation of the BSS is reliant on the vendor of the device installed, however this is seen as an extra feature, rather than a required behaviour. Unfortunately for BACnet, and subsequently the associated people, goods and buildings, BAS vendors have historically been poor at securing their hardware devices and software stack implementations. Additionally, many modern BACnet certified devices operating with the base features defined in previous versions of the protocol standard, further outlined in §4.6.7.1 . Proprietary security measures are being developed, and seems the way forward for securing BACnet managed buildings. Given the market share of BACnet, continuing topology trends, and aim to be integrated into the IoT, additional security features are required to improve the robustness of the protocol.

2.5.1 Device hardening

Not only the BACnet protocol has security issues. As noted, BAS devices have a fraction of the compute power compared to modern day desktops, laptops and smart phones. A number of reported attacks against BACnet managed BASs have used hardware flaws in devices to gain access to the system. One such attack was revealed via the July 2012 Federal Bureau of Investigation (FBI) Newark divisions unclassified situational information report cyber alert, discussing vulnerabilities in the Tridium Niagara ICS system (FBI, 2012). The report detailed that the Niagara system is used extensively in BACnet managed BAS, with at the time, over 300,000 instances operating worldwide (FBI, 2012). The trigger for the investigation was the infiltration of an adversary into a New Jersey air conditioning company, who used Tridium hardware to manage its HVAC system, and supplied the same hardware to other businesses buildings, including financial institutions (FBI, 2012). The Tridium hardware was connected directly to the Internet, with no interposing firewall; a similar situation many businesses face with modern IP connected BAS hardware. Similarly, a hardware flaw in a Tridium controller exposed to the Internet was used by independent researchers to access the BAS in Googles Sydney Wharf building in 2012 (Grubb, 2013). The access was responsibly disclosed, with Google reporting that the system accessed was segregated from other devices on the network. Given the typical topology of BAS, and the topology retrieved in the exposure, the statement is questionable. Analysis outlined in §4.2 found over 5000 unique Tridium devices directly accessible to the Internet over a three year period.

An approach to improving the security of BACnet devices is firmware patching. Similarly, software patching can be used to reduce software vulnerabilities running on each device. However, patching is an area where IT and OT have opposing goals. Patching is a part of the system life-cycle in IT systems, where upon detection of a flaw, a fix is designed, tested and applied to the system. To deploy the patch, often the system must be restarted, or taken offline for a period of time. While appropriate for IT systems, downtime is not acceptable in BASs without forward planning; similar to other control systems which have a safety to security relationship (Dussel et al., 2010; Fisk, 2012; Cheminod et al., 2013). Additionally, patching a vendor installed device often leads to voiding the maintenance warranty of the device, which is a prohibitive associated cost and justification to not patch devices. Much like other control systems, the importance of availability has resulted in a lack of patching, which in turn results in flawed systems in operation for often the entire lifecycle of the device, typically 10-20 years. However noted by Fisk (2012) for longevity of a system, patching must be a part of the system. Fisk (2012) elaborates, stating that patching only becomes effective when an equilibrium is reached where more or equal vulnerabilities are fixed compared to those introduced by the patch. The possibility of introducing unknown error provides further reasoning for BAS to have infrequent to non-existent patching cycles.

While all BACnet devices are tested by a BACnet testing laboratory before being standardised and receiving accreditation, the hardware level security of the device is not a component of accreditation. Rather, the focus is on ensuring a minimum working set of communication and device representation based on specific versions of the standard (BTL, 2017).

2.5.2 Network level security

Khaund (2015) identifies the requirement of a defence in depth, or layered approach to BAS security, including identity validation, firewalls and encryption. BACnet specific network security methods, such as firewalls have been investigated previously in Holmberg, Bender and Galler (2006). Pan, Hariri and Al-Nashif (2014) investigated a means of preventing intrusions automatically through dropping specific packets, akin to a firewall. However in life-safety critical systems, as elaborated by Kaur et al. (2015), dropping packets is not an appropriate action. Comparatively, Fovino, Coletta, Carcano and Masera (2012) implemented a ModBus firewall, which sits between the master and slave devices on a network to monitor the critical state of the system. Alerts are generated based on legitimate commands which could change the state of the system to a defined critical state. Fovino et al. (2012) note that a great deal of knowledge is required of the system to define the critical states, with future work proposed as a means of automatically searching the system space and generating critical state definitions. A similar approach could be used in BACnet/IP, however, it would suffer from similar issues due to the scope of BAS networks, and the multiple interactions between devices. A potential means of enumerating the network protocol could be the use of fuzzing (Takanen, Demott & Miller, 2008). Kaur et al. (2015) undertake fuzzing using a *Python Scapy* based fuzzer to test their network normalisation rules. Further, fuzzing can also be used for improving the security of the protocol, rather than testing a security tool, as noted by (Turner, 2016). Turner (2016) applied a customised fuzzer against the *BACnet open stack v 0.8.4* (Karg, 2016), revealing a buffer overflow in the NPDU implementation. Fuzzing however would be highly time consuming, and have a high degree of risk if running against a real system.

Most BACnet implementations do not have network level visibility of device communication (Caselli, 2015; Jabado, 2017; Humphries, 2017). Rather, a Human Machine Interface (HMI) workstation provides a graphical interface for viewing and controlling specific devices in the network. Further, in Khaund (2015), network visibility of the BAS network is not mentioned as a security countermeasure. The convergence of OT and IT systems has led to data visibility being investigated to further optimise the efficiency of BAS systems, typically for retrieving further knowledge from data, such as occupancy trends from sensor data. With multiple motivations, and limited impact on existing infrastructures, intrusion detection systems are an appropriate avenue to explore.

2.5.3 Intrusion Detection Systems

Intrusion Detection Systems (IDS) are an area of interest in control system network research, and it is no different in BACnet managed BAS. Over the past decade, a range of novel detection approaches have been undertaken to address security issues in BACnet. A Flow based intrusion detection approach was investigated by Čeleda, Krejčí and Krmíček (2012), based on their previous work in Krejčí, Čeleda and Dobrovolný (2012). The flow method, described in Krejčí et al. (2012) was implemented in Čeleda et al. (2012) for IDS purposes on the Masarysk University BAS network, with three security use cases discussed. One use case investigated three BACnet specific attacks, namely a BACnet router spoofing attack, a BACnet DoS and a BACnet write attack. The flow based approach was deemed sufficient by the authors to identify attacks where a combination of packets causes a malicious action, as is the case with the router spoof attack and the DoS. The flow patterns revealed by the write packets however, could not be used to reveal a specific attack taking place, as one packet could cause a malicious write, and thus the pattern is much more difficult to find with the flow based method. A significant limitation of the research is the lack of real attack data, the flows did not reveal any BACnet specific network attacks, in addition, as the monitored network was a real network, attacks could not be crafted and sent across the network to identify what a malicious flow would encompass. The study by Krejčí et al. (2012) identified diurnal and weekly patterns in the traffic of a university BAS. Explained by the purpose of a BAS to control the temperature of an environment based on internal and external temperature differences, which are diurnal. Of note is the diurnal pattern of *Read property*, *Write property*, *I-am* and *Who-is* packets, which form the basis of BACnet communications between devices. These features could be used for deriving normal network behaviour.

Pan et al. (2014) expanded the flow based method of Čeleda et al. (2012) by applying a capture method to a simulated BACnetwork, with a range of BACnet specific network attacks used to generate a dataset for anomaly detection. The methods presented by Pan et al. (2014) assume that the attacker will cause abnormal network transactions to occur, and thus defined normal value ranges for write values. A tuning phase would be required for this method to work, as application to a real network would require additional training to define the normal value ranges of specific devices, as these ranges will be application specific, rather than a global standard. Of the commands classified, the authors classified *I-am* and *Who-is* network transactions as anomalous, however these transactions are normal traffic interactions which occur in a BACnetwork when a device wishes to find another device in the network, and write or read data to said device. Given the conditions

set by the authors, the abnormal method is quite successful, however a limitation of the work is the ability to correctly classify attacks which occur in the bounds of the specification of BACnet. For example, in regards to the classification of *I-am* and *who-is* transactions as malicious, they can be deemed as dual-purpose commands which can cause legitimate-yet-malicious commands to be executed. Similarly, *Write property* commands which send values that are in-bound but with an additional feature, such as frequency or multitude can also cause malicious action.
An alternative method for detection of out-of-bounds attacks is presented by Caselli et al. (2016), using BACnet Protocol Implementation Compliance Statements (PICS) to derive device conformance rules to form IDS rules. The approach provides a semantic-based method to detect abnormal BACnet properties, services and objects operating in a BACnetwork with high detection rates and low false positives. However, similar to Pan et al. (2014), the semantic based approach offered in Caselli et al. (2016) also fails to detect legitimate-yet-malicious commands.

Esquivel-Vargas et al. (2017) continues the work discussed in Caselli et al. (2016), providing rigorous testing to the specification mining approach. A parsing method is developed for BACnet protocol implementation compliance statements for the specific devices operating in the network of interest, which are used to form intrusion detection rulesets. The rulesets are tested using a real BACnetwork, consisting of 646 devices. 10 PICS files describe 640 of these devices, with network fingerprints generated for which services and properties each device has. When traffic deviates from these generated fingerprints, an alert is triggered. The implementation is tested against a long term capture of the real-network, which identified a number of implementation specific anomalous behaviour, and a small testbed used to generate malicious BACnet specific attacks for testing. The results are promising, but as noted by the authors, highly dependent on extraction of device PICS, and observations of normal traffic interactions. Further, as noted, the drawback of this approach is the inability to detect attacks which follow the correct syntax of the protocol and interact with the defined objects and properties (Esquivel-Vargas et al., 2017).
A group of researchers have focused on anomaly based detection approaches to BACnet IDS, carrying on from previous works with the KNX protocol. Kaur et al. (2015) discuss a snort based traffic normalisation method for improving application reliability and security, with future work slated for detection and prevention of attacks. Kaur et al. (2015) first derives a range of attacks from Holmberg (2003), and defines that normal traffic, through real device testing is capable of processing only 180 messages per second. The purpose of traffic normalisation is to improve the robustness of the protocol implementation, through preventing malformed or, out-of-bounds traffic from entering the network. Evaluation of the derived normalisation method is undertaken using a simulated BACnetwork implemented as three virtual machines. The testbed consisted of an attacker which uses the aforementioned fuzzer, the protocol normaliser and one BACnet device operating the *BACnet open stack* (Karg, 2015) and *wireshark* to monitor the network traffic. A range of scenarios were undertaken to generate normal and abnormal traffic sets. Of note, a DoS attack consisting of upwards of 800,000 malformed packets per second was simulated, and used as attack traffic for testing. When the normaliser is deployed, the DoS fails as all out-of-bounds traffic

is dropped by the normaliser, and thus does not reach the device. The traffic normaliser defined in Kaur et al. (2015) achieves the same purpose as Pan et al. (2014) and Caselli et al. (2016), albeit in a different way. Implementation of the traffic normaliser in a real network would be difficult, given the peer-to-peer topology of BACnetworks, as such a number of normalisers would be required. Further work defined by Kaur et al. (2015) is the implementation of a state based classification, similar to work outlined in Peacock and Johnstone (2014) and Johnstone et al. (2015).

Tonejc, Kaur, Karsten and Wendzel (2015) introduced a visualisation method for identifying application layer anomalies in BACnet based on network messages flows. The authors test their method with the premise of hardware malfunctions. Later, in Tonejc et al. (2016), the authors expand and compare a range of unsupervised machine learning methods to the flow visualisation method defined in Tonejc et al. (2015) for identifying malicious anomalies. A limitation of the flow visualisation technique, as represented in Tonejc et al. (2016) is that only previously unseen nodes would be classed as malicious nodes. There is no discussion on how existing trusted nodes who sporadically exhibit malicious action would be identified. Community detection may be an appropriate avenue to pursue, for dormant malicious actors on the network.

While work has been undertaken for intrusion detection in BACnet, current works can be improved. Further work can be directed to identifying contextual anomalies, those which have the characteristics and obey the rules of the protocol, but can still cause malicious outcomes. Detection of contextual anomalies requires learning and understanding the structure of the devices and interactions on the network, baselining behaviour, and identifying where behaviour deviations occur. A common way to find patterns in networks is the application of machine learning algorithms. A typical use case for machine learning in cyber security is in the area of intrusion detection systems. The application of machine learning to BAS's, particularly BACnet, is a recent research direction. As such, a discussion of machine learning applied to cyber security generally is appropriate, to outline areas in which machine learning can be applied to BAS intrusion detection.

2.6 Machine learning

Bhattacharyya and Kalita (2014) define machine learning as the application of algorithms to extract meaningful patterns from datasets, with the intention of applying these learnt patterns to classify future data. There are thus two sides to machine learning, developing learning algorithms, and the application of said algorithms to specific problems. The application of a machine learning algorithm essentially defines a learnt model of behaviour for the system of interest, from which future data is compared, and classified based on the model. There are many learning algorithms, and the classification of machine learning algorithms into specific methods/classes vary between texts, with many crossovers, reclassifications and sub-classifications. Additionally, due to the quantity of algorithms and derivatives, any listing would inevitably be incomplete (Goldstein & Uchida, 2016). There are generally however, two simplified classifications, supervised and unsupervised. The core classifier for placement in these two defined groups is that supervised methods require labeled data, whereas unsupervised methods do not require labeled data. Supervised methods model the relationships between inputs and outputs through analysis of the mapping between inputs and outputs. While unsupervised learning is focused on finding patterns in data sets, and forming clusters from which classification of future data can be undertaken through comparing the distance between new data, and defined clusters. Examples of supervised machine learning methods include Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), decision trees, bayesian networks, and Markov models (Dua & Du, 2011; Bhattacharyya & Kalita, 2014; Goldstein & Uchida, 2016). Unsupervised methods include clustering and statistical methods such as expectation-maximisation algorithms (Dua & Du, 2011; Bhattacharyya & Kalita, 2014; Goldstein & Uchida, 2016).

2.6.1 Application of machine learning to cyber security

A range of machine learning algorithms have been applied to cyber security problems, most commonly for intrusion detection systems (Dua & Du, 2011; Bhattacharyya & Kalita, 2014). Bhattacharyya and Kalita (2014) note the different potential uses for the two generic divisions of machine learning, supervised and unsupervised in network analysis. Supervised machine learning are useful for finding known instances, however they require a labelled dataset which often requires human intervention. Comparatively, unsupervised methods are useful for identifying unknown instances, as they do not require labelled datasets, rather, using the statistical properties of instances to group them by similarity. However, identification of unknown instances is dependent on three core assumptions outlined by Bhattacharyya and Kalita (2014) .

1. Normal traffic has a higher occurrence than Anomalous traffic
2. Anomalous traffic is qualitatively different to Normal traffic
3. Similarity between Anomalous traffic is stronger than between Normal traffic

The features which distinguish between normal and anomalous may cause significant overhead. A core premise of machine learning in cyber security is optimising feature selection to identify the least number of features which can classify the traffic between normal and anomalous with the highest accuracy levels (Bhattacharyya & Kalita, 2014). Through optimisation of features, the subsequent dimensionality of the data is reduced which increases the processing speed of the method (Bhattacharyya & Kalita, 2014).

A major problem with anomaly detection is the ability to detect threats in real-time, to increase the speed in presenting an alarm. Chandola, Banerjee and Kumar (2012) highlight that Markovian based techniques are very effective at undertaking real-time detection (termed online-detection in Chandola et al. (2012)) through slight adaptation, with less than n-order Markovian models applying thresholding, and Hidden Markov Models modifying the forward backward algorithm to compute the optimal state sequence for an observation sequence in real time.

2.6.2 Machine learning in BAS

Tonejc et al. (2016) undertakes a range of unsupervised machine learning methods, including clustering, random forest, one class SVM and support vector classifier. The authors use a test set of known attacks, and pre-process the packet captures into a vector representation, which then has principal component analysis (PCA) applied for feature reduction. PCA can be used to classify network traffic, whereby the lowest number of components are selected which describe the largest amount of network traffic (Dua & Du, 2011). With the assumption that normal traffic outnumbers malicious traffic, at a certain selected number of components, the traffic which is not explained by the components are stated as anomalous. Tonejc et al. (2016) analysis is undertaken at a global level, analysing all traffic on a testbed network. Any cluster smaller than 2% of total traffic is classed as anomalous. Similar to other works in the area, many of the attacks discussed are out of bounds based, e.g incorrect bit combinations, longer than normal addresses. These types of attacks work well for methods such as clustering for outlier detection. The clustering and random forest approaches undertaken were reported as successful, results for the one class SVM were deemed not useful and thus not presented (Tonejc et al., 2016).

As reported by Tonejc et al. (2016), unsupervised methods seem appropriate for detecting anomalies in BAS where the malicious traffic is distinctly different from the normal traffic. However, when detecting anomalous traffic which is close to normal, it is unclear how well unsupervised methods will operate. The requirement of labelled data may be necessary, where supervised methods can be utilised. Typical machine learning methods which operate with labelled data include Markov models and ANNs, which will be discussed in the forthcoming sections.

2.7 Markov Models

A Markov model is a class of probability model which exhibits the Markov property, namely, the determination of the next value is only dependent directly on the current value, and not on historical events or values (Van Mieghem, 2009). Markov models which evaluate based on the current state are called first order Markovian models. N order models, use additional states to assess the next state to transition to, for example, a second order Markovian model would evaluate a pair of states for the next subsequent state. The class of Markov model is dependent on if the state variable is observable, and if the system changes based on the observations from the model. When the state variable is observable, and does not alter a system, the model is classed as a Markov Chain. Alternatively, when the state variable is only partially observable, and the system is not altered, the model is classed Hidden Markov Model. When the system is directly acted upon by the Markov model, it is classed Markov Decision Process; which can be fully observed, or partially observed. Given the domain of this research, only models which do not interact with a system are evaluated, namely, Markov chains and hidden Markov models.

2.7.1 Markov Chains

The state space and time parameter of a Markov chain can differ, and provides a number of further classified Markov chains. When the state space is restricted, a model which exhibits the Markov property is called a Markov chain. There is little consensus in the literature about the naming convention for specific types of Markov chains, with Markov chain or Markov process being used generically to refer to different time parameter models. For this research, if using discrete time, a Markov model is referred to as a discrete-time Markov chain. When using continuous time, a Markov model is referred to as a continuous-time Markov chain.

A Markov chain consists of a set of states, a transition matrix between the set of states and an initial distribution of transition matrix values. A discrete-time Markov chain can be described by Equation 2.1 - Equation 2.4, adapted from Haykin (2009, p583). Equation 2.1 defines the Markov property, while a transition matrix where transitions occur at the same period is defined in Equation 2.2. When Equation 2.3 is met, the transition matrix is called a stochastic matrix. Additionally, Equation 2.4 states that at a discrete time, there occurs a transition in the Markov chain.

Markov chains have a number of properties, including reducibility, periodicity, transience and recurrence. From these properties, the ergodicity, and steady-state analysis of the Markov model can be determined. Reducibility describes the ability of the chain to traverse between states. A Markov chain is classified as irreducible so the chain can get to any state from any state. The periodicity describes the number of periods required to return to the current state from future states. If the period is 1, the state is classed aperiodic, if all states are aperiodic, the chain is classed as aperiodic. If the period is 1<, the chain is classed periodic with a stated period value corresponding to the greatest common divisor of periods between states. Transience and recurrence are properties of states in the Markov chain. A transient state describes a state which may never be transitioned back to, while recurrent states will always, eventually return to the state. Additionally, if a state can never be exited, the state is classed an absorbing state. A state is called ergodic if the state is both aperiodic and positive recurrent. Classification of states, and Markov chains can be determined through analysis of the state properties, and consultation of Figure 2.4. The steady state distribution, sometimes called a stationary distribution, is reached when the probability distribution remains unchanged in the Markov chain as time progresses, stated as a vector π. The steady state is determined by raising the transition matrix to the power k, a future time step, until the transition probabilities converge.

$$P(X_{k+1} = x_{k+1} | X_0 = x_0, ..., X_k = x_k) = P(X_{k+1} = x_{k+1} | X_k = x_k) \qquad \text{(Equation 2.1)}$$

$$p_{ij} \quad = P(X_{k+1} = j | X_k = i) \qquad \text{(Equation 2.2)}$$

$p_{ij} \geq 0$ for all i, j (Equation 2.3)

$$\sum_{j=1}^{N} p_{ij} = 1$$ (Equation 2.4)

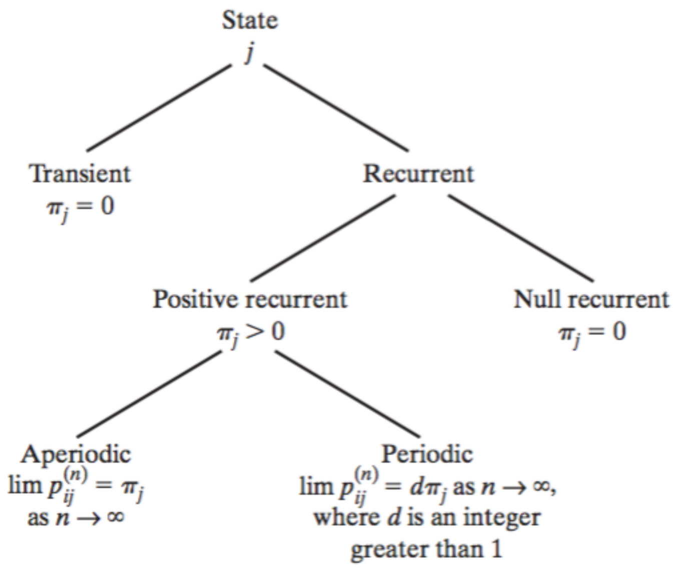

State
j
Transient
$\pi_j = 0$
Recurrent
Positive recurrent
$\pi_j > 0$
Null recurrent
$\pi_j = 0$
Aperiodic
$\lim p_{ij}^{(n)} = \pi_j$ as $n \to \infty$
Periodic
$\lim p_{ij}^{(n)} = d\pi_j$ as $n \to \infty$, where d is an integer greater than 1

Figure 2.4: Markov state classification represented as a decision tree, replicated from Haykin (2009, p590)

Conversely, a continuous-time Markov chain, holds the Markov property, see Equation 2.5, and also holds the concept of time spent in a state, i.e the Δt between transitions are not equal, see Equation 2.6 taken from Van Mieghem (2009). Continuous time Markov chains thus have an exponentially distributed transition time between states, which is independent of changing state; the time spent in the state is not determinant on transitioning to the next state. Both discrete-time, and continuous-time Markov chains can additionally hold the concept of homogeneous and inhomogeneous time, whereby the transition probabilities between states are independent on time, or are dependent on time, respectively. For a continuous time Markov chain, Equation 2.5 describes the Markov property and Equation 2.6 describes the transition matrix. Equation 2.7 is called the Chapman-Kolmogorov equation, which is the fundamental equation determining that states in the set are connected to each other. The same relation as discrete time Markov chains holds for continuous time Markov chains, namely Equation 2.8 stating that at any point in time, the chain will be in a defined state. Further, the initial condition of the transition matrix is Equation 2.9.

$$P(X(t+\tau)=j|X(\tau)=i, X(u)=x(u), 0 \leq u < \tau) = P(X(t+\tau)=j|X(\tau)=i) \quad \text{(Equation 2.5)}$$

$$\begin{aligned} &p_{ij}(t) = P(X(t+\tau)=j|X(\tau)=i) = P(X(t)=j|X(0)=i) \\ &\text{where} \\ &X(t), t \geq 0 \end{aligned} \quad \text{(Equation 2.6)}$$

$$P(t+u) = P(u)P(t) = P(t)P(u) \quad \text{(Equation 2.7)}$$

$$\sum_{j=1}^{N} P_{ij}(t) = 1 \quad \text{(Equation 2.8)}$$

$$\begin{aligned} &P(0) = I \\ &\text{where} \\ &P(0) = \lim t \downarrow 0 P(t) \end{aligned} \quad \text{(Equation 2.9)}$$

2.7.1.1 Markov Chain applications to cyber security

Markov chain models have been used in a range of disciplines including cyber security, typically for network anomaly detection. Markov models can be used for simulation, testing approaches, and classification of new data based on previously seen data. In Caselli, Zambon and Kargl (2015), a discrete time Markov chain is defined for network communication in the ModBus protocol. The model uses a sequence of event process, which allows for state classification based on the sequence of commands being sent on the network, in addition to the data sent on the network. Of note is the generation of the model being independent of the protocol of interest, where each sequence event is automatically classified for state learning. After learning, test data is provided whereby new states and transition patterns are flagged as anomalous, and investigated for further analysis. The concept of weights are applied to commands based on domain knowledge and an adversary model, where not all events are of equal importance.

In Abraham and Nair (2015), Markov models are applied to threat intelligence analytics, through the generation of automated attack graphs from CVSS data. The inputs for this model are the network topology, services running on devices, and CVSS scores of identified devices and software. Markov chains are used to simulate progress of an adversary progressing through the network by chaining vulnerabilities in each device together to reach an end goal. The aim of the research is to define the shortest path an adversary must take to reach their goal, and provide insight into the current security structure of a given system. The model is generic, and could be expanded with the combination of an adversary model, and additional goal states for the adversary.

In Kostakos, Ferreira, Goncalves and Hosio (2016), a Markov chain is developed for prediction of smart phone screen usage, based on statistical properties of usage. The aim of the modelling was to reveal emergent system behaviour, based on the assumption that observable state features are mutually exclusive, and have a probabilistic transition between states. From the data used in this research, the time between sequential events can be determined, defining how long each state is visited for, a continuous time Markov chain. Analysis performed in Kostakos et al. (2016) details that the data exhibits time inhomogeneous features, in certain states the transition probability increases based on the amount of time spent in the current state. Time data analysis of the relationships between state transitions and the time of day and day of week was undertaken. Kostakos et al. (2016) also investigate individual device profiling, applying the analysis to individual devices to identify device specific behaviour patterns for time between states, and time of day state change. A difficulty with time inhomogeneous and continuous Markov models are the increased complexity of the model requiring additional processing capability.

Abaid, Sarkar, Kaafar and Jha (2016) apply a limited state discrete time Markov chain model for detection of botnet infections, and prediction of a botnet attack executing. The model presented defines four states: exploit, binary download, CNC communication and attack. Of note, the authors discuss self loops in their states, where for attack and CNC communication, the transition probabilities are very high (90%<), which significantly impacts the prediction of state change in their model. The authors argue that given their interest in entering an attack state, and not the duration held within the attack state, the self loop can be discounted. From this point, the authors investigate the potential of eliminating all self loops, and conclude that from their dataset, the variance in temporal patterns on self loops is low enough to eliminate self loops completely. The authors describe a high prediction rate for the attack state, and CNC communication state, but detail a limitation of their research is the small sample size and high variance in some state data, with two orders of magnitude more data for attack and CNC states compared to exploit and binary download states. The research proposes a future improved state model, which holds more states, and an improved classifier for classifying data into states.

Bockholt (2017) apply a Markov chain model to identify sequence attacks on network flows from the DARPA98 dataset. The Sequential Probability ratio test and log likelihood ratio algorithms are applied against the Markov chain transition matrix and graphed, with the premise of visually identifying different classes. A linear threshold is implemented which distinguishes the classes of traffic, future work suggests a non-linear threshold. A number of restrictions are placed on the implementation, such as the aim to use low level information, and minimise the processing requirements.

An underlying theme stated in the analysed literature is the requirement of ground truth data representing the normal network behaviours of the system of interest. This is particularly difficult in real networks, where the certainty of having no adversary traffic in the network is unknown. Further, the requirement of labelled data is a time-consuming, often manual process (Bhattacharyya & Kalita, 2014), which must be overcome. An alternative, when the states are not directly observed, and thus not labelled is the Hidden Markov Model.

2.7.2 Hidden Markov Models

A Hidden Markov Model (HMM) is a model which explains a set of unobservable states through a probability distribution function of observations generated by the underlying states (Rabiner, 1990). The observations infer the underlying state of the system, and can be discrete or continuous values. In a network context, the observations could be features of network packets or flows such as packet size, and the hidden state is a discrete value, such as a command, or protocol type (Dainotti, Pescapé, Rossi, Palmieri & Ventre, 2008). The HMM is described by a finite set of hidden states (S), and a finite set of observations (O), generated by those states. The transition matrix (A) represents the probability of moving from one state to another, while the emission matrix (B) describes the probability of the observation when the system is in a specific state. The model is initialised by a starting probability distribution function (π), which defines the probability of starting in each hidden state. As the state is not observed, the joint distribution of the observation and state is taken over the full set of states. Each observation is independent and identically distributed over the state sequence. Each state is dependent on the previous state at the last time slice.

Formally a HMM (λ) is a set of model parameters (π, A, B), defined by Equation 2.10, Equation 2.11, Equation 2.12, Equation 2.13 and Equation 2.14. The joint distribution probability is described in Equation 2.15.

$$S = \{s_1, s_2, ..., s_n\} \qquad \text{(Equation 2.10)}$$

$$O = \{o_1, o_2, ..., o_m\} \qquad \text{(Equation 2.11)}$$

$$\begin{aligned} &A = \{a_{ij}, 1 \leq i, j \geq N\} \\ &\text{where} \\ &a_{ij} = P(s_t = j | s_{t-1} = i) \\ &a_{ij} \leq 0 \end{aligned} \qquad \text{(Equation 2.12)}$$

$$\begin{aligned}&B = \{b_j(k)\}\\&\text{where}\\&k \in O\\&b_j(k) = P(o_t = k|s_t = j)\end{aligned} \qquad \text{(Equation 2.13)}$$

$$\begin{aligned}&\pi = \{\pi_i\}\\&\text{where}\\&\pi_i = P(q_1 = s_i), 1 \leq i \geq N\end{aligned} \qquad \text{(Equation 2.14)}$$

$$P(O|\lambda) = \pi_{s_1} P(o_1|s_1) \sum \prod_{t=2}^{T} a_{ij} P(o_t|s_t) \qquad \text{(Equation 2.15)}$$

Applying a HMM to real data requires solving three problems.

1. The Evaluation Problem: Determine the probability that the model generated the observation sequence
2. The Decoding Problem: Determine the optimal state sequence which best explains the observation sequence
3. The Learning Problem: Determine the probability of the model producing an observation sequence through tuning the model parameters.

The evaluation problem can be solved using the forwards algorithm. The forwards algorithm is a recursive algorithm for calculating the probability of each observation sequence which ends in a specific state, for all states in the model (Shokhirev, 2010; Rabiner, 1990). Using the forwards algorithm, improves the calculation from order $2T \cdot N^T$ to N^2T. The decoding problem can have multiple solutions, as there can be various definitions of optimal (Rabiner, 1990). When the state space is ergodic, meaning transitions between all states can exist, the forwards-backwards algorithm can be used. However, if the state space is not ergodic, the Viterbi algorithm can be used, which is a form of likelihood maximisation algorithm which finds the single best state sequence for the given observation sequences. The learning problem is the most difficult, as there are multiple model parameters to optimise simultaneously. The Baum-Welch algorithm can be used to select a set of locally maximised model parameters for the observation sequence given the model, using an iterative approach which eventually converges (Rabiner, 1990). It should be noted that there may be multiple locally maximised model parameters existing in the explored space, which may require optimisation (Rabiner, 1990).

2.7.2.1 Hidden Markov Model applications to cyber security

HMMs have traditionally been used in speech recognition processes (Rabiner, 1990). HMMs have also been used for general network traffic analysis in Salamatian and Vaton (2001) and later, Dainotti et al. (2008). In Warrender, Forrest and Pearlmutter (1999), the use of HMM in identifying malicious system call sequences was explored, with high true positive (0.8 to 0.99) and low false positive readings (0.0000 to 0.0005). Warrender et al. (1999) note the overhead of HMMs with intermediate data structures, and conclude that HMMs are dependent on the dataset for usefulness. Jain and Abouzakhar (2012) use HMM for network anomaly detection against the KDDcup 1999 dataset (Hettich & Bay, 1999). Two models are derived, a normal HMM and a malicious HMM, where the observations provided are fed into each model and classified normal or malicious based on the observations fit to the model. The overall detection rate values are calculated by summing the F measure of each model for each TCP service investigated. Nine different classes of service are presented, with overall detection ranging from 0.76 to 0.99. An alternative method to having multiple models is assuming that low probability transitions lead to anomalous states (Bhattacharyya & Kalita, 2014), which may be appropriate in certain situations. Ariu, Giacinto and Perdisci (2007), apply a HMM for application level network traffic analysis to identify anomalous sequences of commands travelling over a network. Ariu et al. (2007) explore the use of varying size symbol dictionaries, splitting the training data, and using 10, 20 and 30 hidden states for their model. Similar to results found in Warrender et al. (1999) for system call analysis, the application of HMM to network sequences has a high true positive and low false positive rate. The high classification rates of HMM in both system call and network traffic warrant investigation for application to BAS network data. Further, the periodic nature of diurnal traffic generation lends itself to the use of a state classification method. HMM will be investigated given the lack of application to BAS traffic in the explored literature.

2.8 Artificial Neural Networks

An alternative machine learning method is the Artificial Neural Network (ANN). As defined by Haykin (2009), an ANN is a parallel distributed processor, which consists of simple processing units (called neurons) that can store experimental knowledge and make it available for future use. Knowledge is acquired through the application of multiple input signals to a neuron. Each input signal has a synaptic weight, which provides distinction of importance between inputs in numerical form. The inputs have a summation and bias function applied taking into account the synaptic weights. The value is then passed to an activation function, which limits the output value to be generated based on the type of activation. There are various learning methods, of which batch and online are prominent (Haykin, 2009). The key distinction between the two learning methods are when the change in synaptic weight occurs in the model. For batch learning, synaptic weights can be changed after all training batches have been iterated, called an epoch. Comparatively, online learning provides the ability to change weights at an example by example level inside each epoch.

Additionally, learning algorithms are used to manipulate various parts of the base ANN model, such as the synaptic weights, and the structure of the neurons in the network.

A commonly used learning algorithm is the back propagation algorithm, which provides feedback to prior nodes in the network, improving learning and classification. The back propagation algorithm will be the focus of this section. Back propagation consists of two additional phases. The forward phase where the synaptic weights of the network are fixed, and the input signal is propagated iteratively through the layers until reaching the output layer. Changes in the network only occur in the activation potentials and output neurons during the forward phase. The second phase is called the backwards phase, whereby an error signal is produced through the comparison of the output and desired result. The resultant error signal is then propagated through the network backwards, to affect the synaptic weightings applied to the network.

Formally, there are five steps involved in back propagation ANNs, summarised below from Haykin (2009).

1. Initialisation: Where, assuming no prior information is available, use a uniform distribution with 0 mean and a variance which caused the standard deviation of the neurons to fall between the linear and standard of the Sigma activation function, to derive the weights and thresholds.

2. Presentation: Present the constructed network an epoch of ordered training examples, for each example, perform the sequence of forwards and backwards algorithms.

3. Forward Computation: The input vector is presented to the input layer node, similarly, the desired response vector is presented to the output layer nodes. Computation of the induced local fields and function signals of the network by proceeding forward through each layer of the network, shown as Equation 2.16. Assuming a sigmoid bias function, the output signal of neuron j in layer l is given as Equation 2.17 If neuron j is in a hidden layer, then Equation 2.18 is used, similarly, if neuron j is in the output layer, then Equation 2.19 is used. The error signal is calculated using Equation 2.20.

4. Backwards computation: The aim of the backwards computation is to find the local gradients shown as Equation 2.21, and then adjust the synaptic weights of the network, shown as Equation 2.22.

5. Iteration: The iteration step involves repeating steps three and four while presenting new epochs of training examples until a stopping criteria is met. The order of presentation of the training examples should be randomised between epochs. As training iteration is increased, the momentum and learning parameters are adjusted, often decreased.

$$v_j^{(l)}(n) = \sum_i w_{ji}^{(l)}(n) y_i^{(l-1)}(n)$$

where

$v_j^{(l)}(n)$ is the induced local field

$y_i^{(l-1)}(n)$ is the output signal of neuron i at the previous layer $(l-1)$ at iteration n

$w_{ji}^{(l)}(n)$ is the weight of neuron j in layer l which is fed from neuron i in layer $(l-1)$

(Equation 2.16)

$$y_j^{(l)} = \phi_j(v_j(n)) \quad \text{(Equation 2.17)}$$

$$y_j^{(0)}(n) = x_j(n)$$

where (Equation 2.18)

$x_j(n)$ is the $j'th$ element of input vector $x(n)$

$$y_j^{(L)} = o_j(n) \quad \text{(Equation 2.19)}$$

$$e_j(n) = d_j(n) - o_j(n)$$

where (Equation 2.20)

$d_j(n$ is the $j'th$ element of the desired response vector $d(x)$

$$\delta_j^{(l)} = \begin{cases} e_j^L(n)\phi_j^i(v_j^L(n)) \text{ for neuron } j \text{ in output layer } L \\ \phi_j'(v_j^{(l)}(n)) \sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n) \text{ for neuron } j \text{ in hidden layer } l \end{cases}$$

(Equation 2.21)

where

ϕ' denotes differentiation with respect to the argument

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha[\Delta w_{ji}^{(l)}(n-1)] + \eta\delta_j^{(l)}(n)y_i^{(l-1)}(n)$$

where

(Equation 2.22)

η is the learning rate parameter

α is the momentum cost

2.8.1 Artificial Neural Network applications to cyber security

ANNs have been used in a range of cyber security areas, most prominently for clustering, feature extraction and anomaly detection in network traffic (Bhattacharyya & Kalita, 2014). Ryan, Lin and Miikkulainen (1998) examined the application of an artificial neural network to *Unix* command detection in one computer with multiple unique users. A classification task was undertaken for normal traffic to identify which user used what commands. In addition, random data was generated to represent anomalous behaviour on the system. The implemented method reported a 0.96 detection rate of anomalous behaviour, with a 0.93 true positive rate. The 0.07 false positive data was explained due to a lack of data for a specific user of the system whose only interactions were the reported false positive data. Ghosh and Schwartzbard (1999) evaluated the DARPA 98 intrusion detection dataset, using an ANN for misuse and anomaly based detection. 139 normal and 22 anomaly sessions were used for testing both approaches. Increasing the accuracy of detection results in a higher false positive rate, with a 0.773 true positive and 0.036 false positive accepted as the optimal outcome for the anomaly detection approach. In regards to network anomaly detection, ANN's have been applied to detect a range of network based attacks (Bhattacharyya & Kalita, 2014). Often, this type of research explores frequency based attacks, such as denial of service and port scanning, whereby the normal network traffic has synthetic attacks embedded into the dataset, such as in Andropov, Guirik, Budko and Budko (2017). Callegari, Giordano and Pagano (2014) evaluated a range of ANNs for prediction of future network traffic for finding anomalous traffic. The results are promising, with false positives ranging from 0.002 to 0.049. With increasing applications of ANNs to anomaly detection in cyber security, application to a BAS network is of interest.

2.9 Summary of literature

With the presented review of the literature, it has been determined that intrusion detection approaches with regards to BACnet/IP managed BASs should be further explored. The majority of research has focused on incorrect traffic identification as anomalous behaviour, such as increased frequencies or illegal byte values. Further, two distinct types of dataset are used. Either, a real-dataset with synthetic attack traffic implemented, or a small simulated test-bed network with a limited range of attack types. Additionally, implemented test-beds are often not analysed in the context of real-world network implementations.

Few articles investigate the application of machine learning to anomaly detection in BACnetworks. Methods, such as Markov based models and ANNs have seen high accuracy rates in anomaly detection in similar network structures such as SCADA networks. As such, improvements can be made in both the datasets used to test anomaly detection approaches for BACnetworks, and the application of existing machine learning algorithms for anomaly detection in BACnetworks. The study aims to address these issues, through generation of a simulation testbed for anomalous BACnet command generation, and the application of machine learning methods, such as HMMs and ANNs for evaluating the detection capabilities for BACnetworks.

Chapter 3

Research Methodology and Design

This chapter details the methodology and design selected for the research. In some circumstances it is straightforward to identify known-known anomalies in networked computer systems where the variables are well understood and controlled, allowing for the experimental method to be directly applied. However, identifying other types of anomalies, such as those defined in Table 1.2 are not as straightforward to evaluate. To explore this problem, a number of frameworks are evaluated to identify the appropriate research approach that could address the posed research questions.

The dominant research paradigms in information systems and cyber security research are explored, and an appropriate paradigm is identified and justified. The approach taken for the research is discussed, with a range of candidate methods identified, analysed and ultimately selected. Next, the research questions and derived hypotheses are stated. Following, the research design is presented, detailing the phases of research, core variables and materials used. The limitations and threats to research are identified, and subsequently mitigations to said threats are stated. The chapter concludes with a discussion of validity testing within the study.

3.1 Methodology

A methodology, as defined by Williamson (2013) is the "overall logic of inquiry involving the philosophical assumptions behind an inquiry, the strategy of conducting research such as research design and selection, and adoption of research methods and techniques as well as arguments for knowledge construction and justification"(Williamson, 2013, p. 116). From this definition, there are two distinct features when undertaking research. First, the methodology, which defines the philosophical assumptions underpinning the research used to inform the overall strategy and form a foundation for selecting a research design. Second, the research design, which details the methods and techniques, defining specific processes and procedures used for conducting the research, as well as the collection and processing of data. Crotty (1998) states another feature, the justification of using a specific methodology and design, rooted in the research question, and the assumptions of reality the researcher brings to the research. Thus, through exploration of the philosophical assumptions of a research domain, a critical selection process was undertaken for the use of specific methods and techniques which constitute a design. Coupled with analysis of the research questions, justification can be provided for a specific selection. Given the research undertaken involves technology, systems and information, the domain of information systems research is explored for paradigms, and associated meta-theoretical assumptions.

3.1.1 Research paradigms

As noted, the combination of meta-theoretical assumptions can be collectively used to inform a research paradigm. For information systems research, Williamson (2013) defines three dominant paradigms, positivist, interpretivist and critical. Similarly, Galliers (1990), describes the paradigms of research in information systems as a continuum between empirical (positivist) and interpretative (interpretivist) views. While the term positivist and quantitative, and likewise interpretivist and qualitative are often used interchangeably, Williamson (2013), notes they are not synonyms. The terms do not exclusively match with the aforementioned paradigms. Discussed by Crotty (1998), the qualitative and quantitative classifications for research should be applied at the method level, as they classify types of methods, rather than at the methodology level (separated as theoretical perspective and epistemology by Crotty (1998)). This view is also informed through Galliers (1990) classification schema of empirical and interpretative. As each methodology can utilise methods from both classes, quantitative and qualitative. The research paradigms will be discussed in Galliers (1990) and Crotty (1998) terms. Namely, the paradigms are classed in terms of empirical (positivist) and interpretative (interpretivist), while methods are classified as quantitative to qualitative, detailed in Figure 3.1.

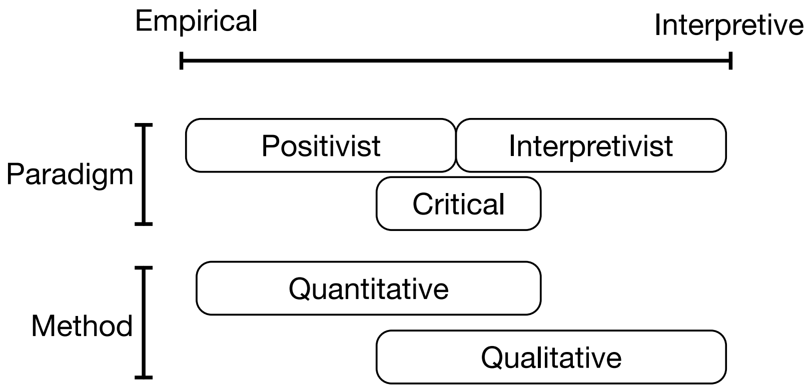


Figure 3.1: The continuum of dominant research paradigms in information systems, and method classifications.

Positivists aim to apply scientific methods to a problem domain, with close association to deductive reasoning and quantitative data. The aim of the positivist is to discover general laws which can be applied, termed nomeothic. Interpretivists are concerned with meanings constructed through individuals and groups, principally using inductive reasoning and collecting qualitative data, termed idiographic. Critical theory focuses on changing the structures of society to empower disadvantaged or minority groups, through quantitative and qualitative data and subsequent methods.

3.1.2 Meta-Theoretical assumptions

When distinguishing between the various paradigms, classification is undertaken through the use of differing meta-theoretical assumptions. In information systems research, there are a number of implied meta-theoretical assumptions which guide the choice of research methodology. Four core meta-theoretical assumptions, derived from Williamson (2013) and Guba and Lincoln (1994) are ontological, epistemological, logic of inquiry and axiological. Crotty (1998) posits that ontological and epistemological issues arise together, whereby a certain way of understanding "what is" (the ontology) is tightly coupled with the way of understanding "what it means to know" (the epistemology). Crotty (1998) argues that it is difficult to separate epistemology and ontology as concepts, as they inform one another, and thus selection of a methodology (termed theoretical perspective) is often complicated through stating the epistemological and ontological assumption. Rather, for the sake of completeness, the methodology discussion presented will follow the specification derived from Williamson (2013) and Guba and Lincoln (1994) stating both the epistemological and ontological assumptions, with knowledge that the two assumptions inform one another. The definitions of each meta-theoretical assumption can be found in Table 3.1.

Table 3.1: Summary of meta-theoretical assumptions, adapted from Williamson (2013), Guba and Lincoln (1994)

Assumption	Description
Ontological	The nature and existence of social reality
Epistemological	The nature of knowledge and the ways of knowing
Logic of Inquiry	The logic of scientific explanation
Axiological	Ethics and claims about values and their impact on research

3.1.2.1 Ontological assumptions

The ontological assumption or question, relates to the form and nature of reality (Williamson, 2013). Through defining the ontological assumption, research questions are grounded in what is perceived to be "real", and thus knowledge can be acquired which describes "how things really are" (Guba & Lincoln, 1994, p. 108). The three dominant paradigms have competing views of the ontological assumption. The positivist view accepts that there is one ordered and stable reality, which exists regardless of interaction, called realism (Guba & Lincoln, 1994; Williamson, 2013). Interpretivists view reality as a social construct, where reality is fluid and exists for the individual or group when experienced, called relativism (Guba & Lincoln, 1994; Williamson, 2013). A middle ground is the critical paradigms view, which accepts that reality objectively exists without interaction, but was shaped through the individual or groups construction over time, and now confines what is real, called historical realism (Guba & Lincoln, 1994).

3.1.2.2 Epistemological assumptions

Epistemology, as stated by Crotty (1998) is "how we know what we know". Epistemology is used to ground research and allow for decisions of what can be classified as knowledge, and how knowledge can be described as adequate and legitimate. A core epistemology is objectivism, which states that meaning exists apart from the operation of consciousness. As an object, of a certain type, there exists an intrinsic meaning defining the object. Thus, when an object is identified as a specific type of object, the meaning of said object is revealed. Objectivity aligns with the positivist paradigm, and, as noted, is clearly aligned with the ontological assumption of realism. Another, competing epistemology is constructionism, which rejects the objectivist view, stating that there is no objective truth waiting to be discovered, meaning exists due to the interaction with the world. Thus, meaning is not discovered, but rather constructed by the individual, who may have competing meanings for the same object of interest. Constructionism aligns with the interpretivist paradigm, and is closely formed by the relativism ontological assumption. A third epistemology called subjectivism aligns with both the interpretivist and critical paradigms, thus being influenced by historical realism. Subjectivism states that knowledge is value mediated, and thus meaning is value dependant and can change (Guba & Lincoln, 1994).

3.1.2.3 Logics of inquiry

There are two dominant logics of inquiry also called logical reasoning, namely, deductive and inductive. Deductive reasoning begins with a general theory about an object of interest, testable hypotheses are formed, and observation is undertaken. The end result of deductive reasoning is an acceptance or rejection of the stated hypothesis, providing suggestion of a theory. Conversely, inductive reasoning begins with a specific observation, which is then explored to find patterns and thus form temporary hypotheses. The hypotheses are then explored and a general conclusion or theory is then suggested. Typically, deductive reasoning aligns with a positivist paradigm, while inductive reasoning aligns with interpretivist paradigms.

3.1.2.4 Axiological assumptions

Axiology is the study of the nature of values, when making an axiological assumption, one is assessing the impact of individuals or a group of people's values which influence the research (Williamson, 2013; Guba & Lincoln, 1994). Positivist paradigm research defines that research is undertaken in a value-free way, with the researcher being independent from the data, allowing an objective stance. Interpretivist paradigm research defines all research as value laden, as the researcher is part of the study, and cannot be separated, thus research is subjective. A summary of the dominant meta-theoretical assumptions in information systems is presented as Table 3.2.

Table 3.2: Summary of meta-theoretical assumptions for the three dominant paradigms in information systems research, adapted from Guba and Lincoln (1994), Crotty (1998), Williamson (2013)

	Paradigms		
	Positivist	Interpretivist	Critical
Reason for Research	Discovery of regularities and causal laws for explanation, prediction and control of events and processes.	Description and understanding of phenomena in the social world and their subsequent meanings in context	Empower people to change their conditions through unmasking and exposing hidden forms of oppression, false belief and commonly held myths.
Ontology	Realism: Ordered and stable reality exists irrespective of an observer	Relativism: Reality is socially constructed, as people experience and assign meaning, making reality fluid and fragile.	Historical Realism: Reality is socially constructed but also perceived as objectively existing.
Epistemology	Objectivism: Instrumental approach to knowledge. Knowledge enables control of events, and represents reality being stable and additive.	Constructionism: Practical approach to knowledge. Inclusion of evidence about a subject, and context is used to empower understanding of other realities, and how the researcher came to understand them.	Subjectivism: Dialectical approach to knowledge. Knowledge reveals hidden forms of control, domination and oppression, which empowers individuals to seek social change and reform.
Logic of Inquiry	Deductive: hypothesised relations among variables, logically derived from laws or theories are empirically tested in a repeatable way.	Inductive: development of idiographic descriptions and explanations based on studies of people, and actions in context. Explanations need to make sense to the individual or group being studied as well as the researchers and subsequent research community.	Deductive, Inductive or Abductive: seeking creative leaps and revealing hidden forces or structures which help people understand their circumstances and ways of changing them.
Axiological	Value-Free: Assumes both natural and social sciences are objective and value free, operating separately from social and power structures. Ideally, positivist researchers are detached from the topic of interest, and collect value-free facts	Value-Laden: Questions the possibility of value-neutral science and value free research. Values are seen as embedded in all human action, and hence are inevitably part of everything studied, without judging one set of values as better than another.	Value-Reasoned: Describes any research as a moral-political and value-based activity. Critical researchers explicitly declare and reflect on their value positions and provide arguments for their normative reasoning.

3.1.3 Selection of methodology

According to Williamson (2013), selection of a research paradigm is closely related to the form of the research question, which infers the values of the core meta-theoretical assumptions. Questions which aim to use measurements which can reliability provide an objective answer, and can be generalised to the population on which the study is based, can derive its assumptions from the positivist paradigm. Questions which do not lend themselves to measurement and objective results can instead derive their assumptions from the interpretivist paradigm.

From analysis of the stated major research question *RQ1: How can known and unknown attacks against BACnet/IP based Building Automation Systems be detected?*. An objective, empirically proven answer would be appropriate, and thus the research paradigm leans towards a positivist approach. A similar method is described by Crotty (1998), whereby a research question or problem informs the method to answer the question, and thus is the foundation of the strategy to undertake subsequently leading to a discussion of assumptions about knowledge and reality; the methodology.

A complementary approach to identifying the meta-theoretical assumptions and subsequent paradigm of research is through the analysis of the object of interest, and associated methods described by Galliers (1990). Analysis of the objects of interest from Galliers' taxonomy, shown as Table 3.3, and the research questions posed suggests that both technology and theory testing are potential subject matters for this research. From the identified objects of interest, the approaches which align with both objects are highlighted, where the classification of "yes" is stated in one of the objects of interest, it is explored. The "possibly" ranked approaches omitted from exploration are Case Study, Survey, Subjective and Descriptive, given the research questions do not lend themselves towards these approaches. The approaches explored are Theorem Proof, Laboratory Experiment, Field Experiment, Forecasting and Futures Research, and simulation and game/role play.

Theorem Proof is described in Galliers (1990) as the development and testing of theorems, closely tied to the strengths of the scientific method, namely, repeatability, reductionism and refutability, in addition to the precision of results. In essence, Theorem Proof is primarily concerned with mathematical theorems, and thus will be discounted as an approach for this research.

Laboratory Experiments are described as the identification of precise relationships between variables in a designed controlled environment, with analysis undertaking using quantitative techniques. With the aim to derive generalist laws to be applied to real world situations, the embodiment of the positivist paradigm. Given the empirical nature of the research questions,laboratory experiments is an appropriate approach.

Field Experiments are an extension of Laboratory Experiments, with the aim of improving the real-world applications of research through some external object of interest, which the researcher has limited control over to experiment on. The lack of control, and repeatability that this approach would incorporate, and potential of damaging real networks with attack traffic discounts this approach.

Forecasting and Futures Research, defined by Galliers (1990) are scientific (positivist) and interpretivist terms for determining future behaviour or action based on past historical data. In a positivist tense, Forecasting involves the use of statistical models, such as regression and time series analysis on numeric data. From an interpretivist perspective, in terms of technology, Futures Research is the interpretation of behaviour over time related to a specific technology. Given the research questions, Forecasting aligns with the operation of intrusion detection systems, and the purpose of the research, namely, to detect future known and unknown threats based on historical threats and an understanding of the system of interest. As such, the positivist approach of Forecasting could be an appropriate approach for this research.

Similarly, Simulation and Game/Role Playing represent the positivist and interpretivist aspects of the approach, being, the extrapolation of a model to solve problems which are difficult to solve analytically from observable behaviour. In terms of Simulation, this is a mathematical, or computer based construct which represents a system of interest. Whereas, for Game/Role Playing, a hypothetical situation is devised and prepared for testing in a real-world scenario. Through the use of Simulation, complex problems can be reduced, explored and extrapolated based on controllable factors. Given that a BAS is a complex system, the ability to simulate parts of the system to answer controllable, repeatable questions is justifiably appropriate for the research. As such, Simulation can also be classed as an appropriate approach.

Through the exploration of the objects of interest, and approaches combined, there exists three potential approaches to undertaking the research, Laboratory Experiments (Experimental), Forecasting, and Simulation. The shared paradigm of these approaches are positivist. Derived from the approaches of Galliers (1990) and Williamson (2013), and reflection upon the research questions, a positivist paradigm has been accepted for the research.

3.1.4 Summary of accepted meta-theoretical assumptions

A realist ontological view is taken for this research, given that the research is based on empirical comparisons of measures, the belief that reality is stable and existent irrespective of observation. An objective epistemology is taken, based upon the acceptance of reality existing irrespective of observation, knowledge is derived through the control of events, to be determined through the analysis of appropriate identified methods. The logic employed for the research follows deductive reasoning, the classical positivist logic, as the research naturally forms from a general theory about anomaly detection in BAS, with room for testable hypotheses to be developed. The role of values in the research follows from the objective epistemology, the researchers values are of no consequence for the exploration, testing, observation and analysis of the research problem posed. A summary of the selected meta-theoretical assumptions are presented in Table 3.4. With the paradigm now apparent, the associated quantitative methods and techniques, relating to the identified approaches, Experimental, Forecasting and Simulation shall be explored to derive a research design.

Table 3.3: Galliers' taxonomy of research approaches and objects of interest, adapted from Galliers (1990)

	Traditionally Empirical approaches					Overlapping approaches		Traditionally Interpretive approaches		
Object of interest	Theorem Proof	Laboratory Experiment	Field Experiment	Case Study	Survey	Forecasting and Futures Research	Simulation and Game /Role playing	Subjective /Argument-ative	Descriptive /Interpretive (Reviews)	Action Research
Society	No	No	Possibly	Possibly	Yes	Yes	Possibly	Yes	Yes	Possibly
Organisation /Group	No	Possibly	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Individual	No	Yes	Yes	Possibly	Possibly	Possibly	Yes	Yes	Yes	Possibly
Technology	Yes	Yes	Yes	No	Possibly	Yes	Yes	Possibly	Possibly	No
Methodology	No	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Theory Building	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Theory Testing	Yes	Yes	Yes	Possibly	Possibly	No	Possibly	No	Possibly	Possibly
Theory Extension	Possibly	Possibly	Possibly	Possibly	Possibly	No	No	No	Possibly	Possibly

Table 3.4: Accepted meta-theoretical assumptions of the research

Assumption	Description
Ontology	Realism
Epistemology	Objectivism
Logic of Inquiry	Deductive
Axiomatic	Value free and separate from power struggles

3.2 Research approach

As identified through the exploration of the methodology, there were three potential approaches appropriate for the research. The forthcoming section describes the methods of the three approaches, Experimental, Simulation and Forecasting.

3.2.1 Experimental approach

The core concepts of experimental methods are the construction of hypotheses, identification of statistical tests, definition of variables, and a discussion of the reliability and validity of the design (Williamson, 2013). Further, experimental methods involve undertaking the experiment according to the design, performing the appropriate statistical analysis and drawing conclusions to suggest recommendations. The experimental method process is visually represented as Figure 3.2, with the concepts discussed in the forthcoming sections.

3.2.1.1 Hypotheses

There are a number of reasons for using an experiment, discussed by Montgomery (2013) and detailed in Table 3.5. The overall objective of an experiment is to.

1. Determine the influential variables against the response variable.
2. Optimise the control variables to ascertain a desired response.
3. Optimise the control variables to reduce the variability of the response.
4. Reduce the effects of uncontrollable variables.

Recognition of an experimental reason is paramount to form appropriate questions, and thus derive hypotheses to form experiments from. A hypotheses, as defined by Williamson (2013), is a statement or proposition about a predicted relationship between two or more variables which is empirically testable. Noted by Montgomery (2013), generally, a single experiment is not adequate to answer a question, thus a set of smaller experiments, of multiple types are used to suggest answers to posed questions. The notion of hypotheses thus allows exploration of "smaller questions", which can be used to inform of a larger overarching question. Once a hypotheses is formed, classification of variables in the object of interest can commence.

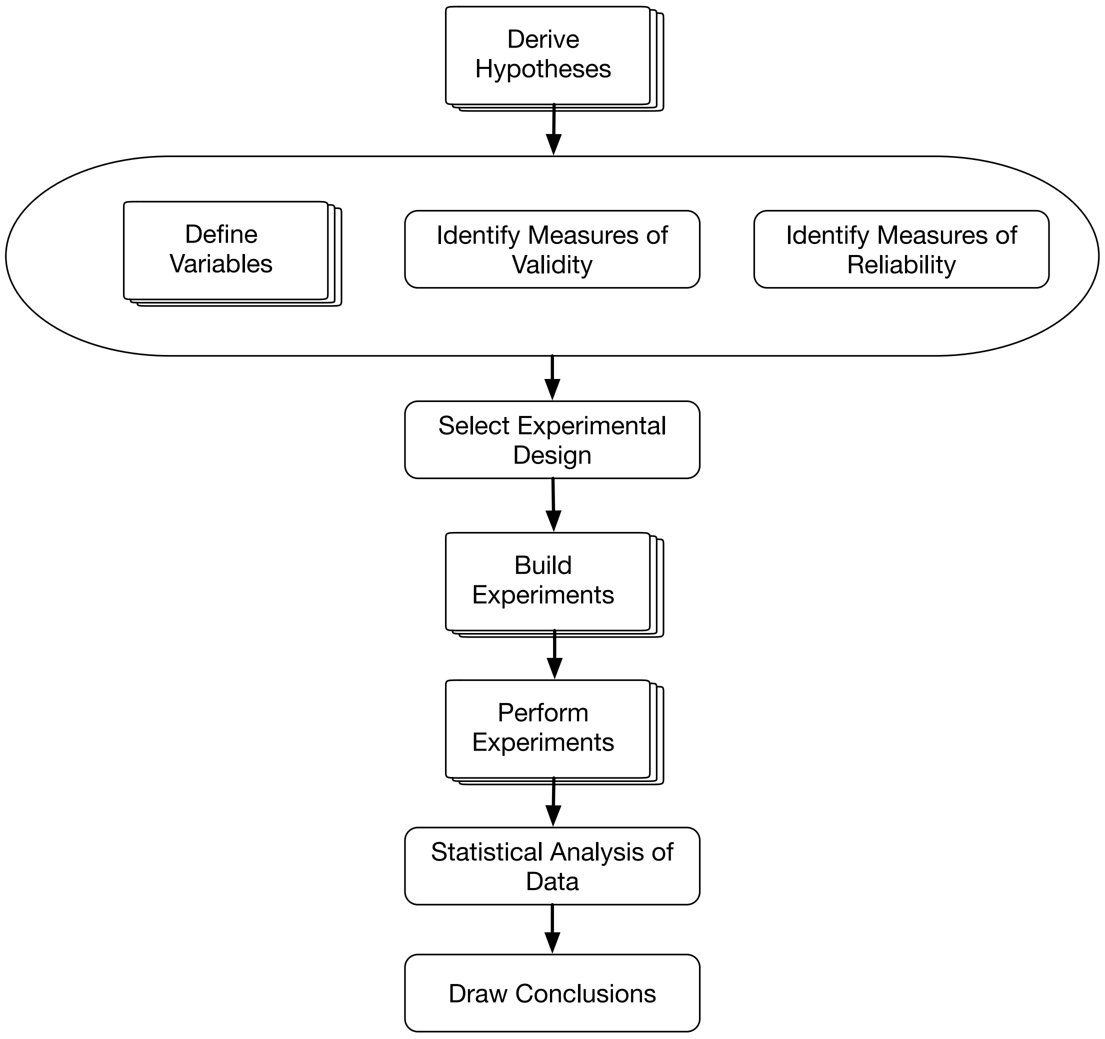

Derive Hypotheses
Define Variables
Identify Measures of Validity
Identify Measures of Reliability
Select Experimental Design
Build Experiments
Perform Experiments
Statistical Analysis of Data
Draw Conclusions

Figure 3.2: Summary of experimental method processes, adapted from Williamson (2013), Montgomery (2013)

3.2.1.2 Variables

Independent variables are the factor which is manipulated to observe what impact change has on other variables. The independent variable is the presumed cause of the phenomena in the object of interest. The dependent variable is the factor which is measured to determine response to a particular change in the independent variable. The control variable is the factor which is not of interest, but is held constant to ensure the control variable does not impact the independent or dependent variables. The confounding variable is an unknown element, which is not the focus of the study, but is assumed to affect some of the observations, and must be accounted for. The extraneous variable is a competing independent variable which may explain outcomes in the dependent variable. The final type of variable is the moderating variable, which moderates the impact, or describes the relationship between other variables in the study.

Table 3.5: Reasons to undertake experiments, adapted from Montgomery (2013)

Experimental Reason	Description
Characterisation	Learning which variables influence the response of interest. Generally used when there is a lack of knowledge about the object of interest.
Optimisation	When the variables have been identified, optimisation experiments are used to tune variables to form a desireable output.
Confirmation	Used to verify behaviour of an object of interest, consistent with a theory or experience.
Discovery	Determine what happens when new variables are explored in an object of interest.
Robustness	Determining in what condition the response variables degrade, to better control variability in the response from poorly controlled cofounding variables

Through identification of the variables and analysis of the hypotheses, an experimental design can be selected.

3.2.1.3 Reliability, validity, and threats

Reliability in terms of an experimental method is the consistency of results produced by a measuring instrument, when repeated in a similar situation. Validity, subsequently, is the capacity of a measurement instrument to measure what is purports to measure, or the accuracy of observations taken by the instrument. Reliability and validity have a co-existent relationship, validity infers reliability, however, reliability does not infer validity, rather, reliability is a determinant of how valid a measure can be. The purpose of ensuring reliability is to reduce the rate of random error in measurements. Mitchell and Jolley (2010) detail three sources of random error, namely, observers, participants and test situation. Determining the extent of random error can be achieved using a test-retest process, whereby multiple runs of an experiment are taken and compared, to determine any change. If change due to random error is determined, additional methods, such as measures of reliability and measures of internal consistency can be used to identify the source of reliability degradation.

It is important to establish the validity in addition to the reliability, as an invalid variable can be reliably measured. There are two types of threats to the validity of an experiment. The internal validity, which is the confidence that observed results are attributable to the independent variable, and not caused by other, unknown factors. External validity is the generalisability of the research findings, or, the extent of application to other populations, settings or treatments. The strength of each type of validity is a distinction between laboratory and field experiments. Laboratory experiments are high in internal validity due to the control provided from the surroundings. Allowing manipulation of the independent variable, accuracy of effect measurement on the dependent variable, and ensuring that the effect of cofounding variables are controlled or mitigated. The generalisation of results, or external validity, is often questioned due to the results being drawn from an environment which may not resemble the “real world”. Conversely, the lack of external control in field experiments questions the internal validity of an experiment, whereas the external validity is generally higher. Elaborated upon by Mitchell and Jolley (2010) and detailed in Table 3.6, there are common threats to validity and reliability faced by laboratory and field experiments, however there are also precautions which can be applied to eliminate these potential threats.

Table 3.6: General threats and precautions to experiments, adapted from Mitchell and Jolley (2010)

Threat	Description	Precaution
History	External events that are mistaken as treatment effects	Isolate participants from external events
Maturation	Biological changes that are mistaken as treatment effects	Conduct study in a short period
Testing	Treatment effects are due to participants having learnt from the pretest	Only test once, use different versions of the test to decrease the testing effect
Instrumentation	Changes in measuring instrument causes treatment effects	Be consistent with materials and measurements
Mortality	Apparent effects are due to participants leaving the study	Use incentives and reduce number of treatments to prevent participants stopping
Regression	Treatment effects are due to extreme scoring results in the pretest being less extreme in the post test	Do not choose participants based on the basis of extreme scores
Selection	Effects are based upon the control groups being different before the study started	Don’t use designs that involve comparing groups of participants with each other.
Selection Interactions	Effects are due to groups that scored similarly in the pretest naturally growing apart and scoring differently in the post test	Match on all relevant variables, not just on pretest scores.

3.2.2 Experimental designs

There are a range of experimental designs, the selection of which is dependent on the hypotheses, the variables, and the object of interest. Additionally, in computer science research, difficulties can arise with the control of non-treatment variables due to the volatile state of computer memory. To control non-treatment variables, the use of quasi-experiments can be undertaken, which limits the effects of non-treatment variables by identifying, and proving that the change in the dependent variable was not caused by changes to the non-treatment variable.

3.2.2.1 Randomised two group design

The basic experiment, or Randomised two-group design involves two identical groups which are treated differently throughout the experimental process. One group termed the treatment group receives a treatment, while the second group receives no treatment, termed the non-treatment or control group. Analysis of these two groups is undertaken, where the treatment or independent variable is the only systematic difference between the groups. To ensure that the treatment is the only systematic difference applied, independent random assignment is used where assignment of participants or objects to each group is randomised. The groups are then measured based on the dependent variable, from which two hypotheses can be defined, namely the experimental hypothesis: the treatment has an effect, and the null hypothesis: the treatment does not have an effect. Additionally, temporal precedence must be established and spuriousness must be identified. Temporal precedence proves that the changes in a group occurs after the treatment variable has been applied. While spuriousness shows that the variation observed between both groups could only be due to the treatment applied. Without temporal precedence, the variable which causes change cannot be determined, similarly without measuring spuriousness causality cannot be determined. Statistical analysis is undertaken on the dependent variable, and conclusions drawn. The basic model can be varied to include alternative treatments, rather than a control group, with comparison between each alternative treatment, and no treatment.

3.2.2.2 Pre-test/Post-test control group design

A pre-test, post-test control group design, often called a covariance design is similar in premise to the randomised two group experimental design. The difference being that both groups are initially tested before a treatment is applied. After initial testing, the treatment group undertakes a treatment and is then measured. The control group is isolated from the treatment group, and tested at the same time as the treatment groups post-test. Observations are then derived from, and attributed to the change, or lack of change between the treatment and control group.

3.2.2.3 Factorial design

When there exists multiple independent variables in a study, both the independent effects, and the interactive effects between the variables on the dependent variable is studied, termed Factorial design. Unlike experiments which evaluate independent variables individually, factorial designs allow for combinations of independent variables to be explored jointly, revealing potential effects on the dependent variable caused via interactions between independent variables. Confounding variables can also be built into the design, providing a level of control not offered in some other experimental designs. Factorial designs can also reduce the number of observations required when compared to other experimental designs; the number of observations required for a factorial design is determined through power analysis. A drawback of Factorial design is the increased complexity as the number of independent variables increases.

3.2.3 Quasi-Experimental Designs

Quasi-experiments have a similar aim as true experiments, where a treatment is administered. However, when using a quasi-experiment design, treatment and control group participants are not randomly assigned. In addition, non-treatment factors are identified and taken into account, rather than eliminated. Quasi-experiments aim to establish temporal precedence in the same manner as true experiments, covariation is also assessed through comparison of treatment and non treatment conditions. However, spuriousness through randomising the group values cannot be eliminated, nor are non-treatment factors kept constant to account for spuriousness. Rather, non-treatment factors are identified, and explored to justify that the observed change is not due to a non-treatment factor. The eight general threats to validity, defined in Table 3.6 can be used to eliminate specific threats and infer the causality in the dependent variable. Each quasi-experimental design eliminates different threats, with remaining threats requiring additional controls to eliminate, or explain the impact of the threat, explained in §3.2.3.1, §3.2.3.2, §3.2.3.3, and summarised in Table 3.7.

Table 3.7: Threats to internal validity mitigated by stated quasi-experimental designs, adapted from Trochim (2006), Mitchell and Jolley (2010), Williamson (2013), Edgar and Manz (2017)

Potential Threats	NEGD	ITSD	RD-D
History	✓	✓	✗
Maturation	✓	✗*	✗
Testing	✓	✗*	✗
Instrumentation	✓	✗*	✗
Mortality	✓	✗*	✗
Regression	✓	✓	✗
Selection	✗	✗	✗
Selection Interaction	✗	✗	✗

* Requires additional controls to mitigate

3.2.3.1 Non-Equivalent group design

The non-equivalent group design (NEGD) is a direct comparison to the simple experiment, as noted by Trochim (2006), Mitchell and Jolley (2010). The NEGD is structured as as pre-test/post-test randomised experiment, with the randomised assignment removed. Groups of subjects are formed on the notion of similarity, however, true certainty of the groups comparisons cannot be stated, given the threats to internal validity faced. The selection threat is not accounted for in the design due to the deliberate group separation. The interaction between threats and selection, termed selection interaction in Mitchell and Jolley (2010) is dependent on the data analysed, and thus can be explained given the pre and post bivariate data. Mitchell and Jolley (2010) states that the NEGD has all the strengths of a simple experiment, with the design eliminating the threats of maturation, testing and instrumentation. While the use of a control group allows for the explanation of history, maturation and mortality threats. Further controls, such as variable matching or statistical techniques can assist in reducing the impact of selection interaction threats (Christensen, Johnson & Turner, 2011)

3.2.3.2 Interrupted time series design

An interrupted time series design (ITSD) uses a series of observations and/or measurements over an extended period of time, before and after the treatment condition, rather than two distinct pre and post tests. When the point at which a treatment occurs, the "interruption" treatment can be identified through the change in the slope when plotting the observations (Shadish, Cook & Campbell, 2002). The design allows for trends to be examined before, during and after a treatment, testing and re-testing is also performed on the same subjects, thus, selection and selection-interaction threats to internal validity are eliminated by design. Conclusions drawn in ITSD are only valid if the effects of history, maturation, mortality, testing and instrumentation during the study's time is estimated correctly. The greatest threat to internal validity is that of history, as the precaution is at odds to the design. An alternative measure for ITSD could be the collection of baseline data to account for historical effects, which could reveal cyclical patterns, which, without a historical evaluation, may be mistaken for treatment effects. In addition, any inconsistent effect could be a threat to internal validity, specifically for history and regression, however, inconsistency in maturation, testing, mortality and instrumentation are possible, but can often be estimated, thus being accounted for in the design (Mitchell & Jolley, 2010). Where estimation is inaccurate, ITSD should attempt to eliminate the threat, rather than account for it, through similar measures in randomised experimental designs.

3.2.3.3 Regression-discontinuity design

The regression discontinuity design (RD-D), is similar to NEGD, with a pre and post test, however, there are two (or more) groups, whose selection is based on a continuous numeric criterion derived from the pre-test. In this way, groups of interest within specific value ranges determined by the pre-test, are given treatments, while other classed groups are set as control groups. A regression line is fit between both the treatment and control groups, at the point of the cutoff, a discontinuity is present, which can show the effect of the treatment, as opposed to the control group. The RD-D counters many threats to internal validity faced by other quasi-experimental designs, and thus can be comparable to the validity of fully random experimental designs Trochim (2006).

Rather than attempting to match variables or prove the equality of groups pre-test to infer post-test results, the RD-D assumes that without a treatment, the pre-post relationship between the two groups would be identical. The strength of the RD-D is thus dependent on two further assumptions, that there is no spurious discontinuity in pre-post relationship which occurs at the cutoff point, and the degree of validity in the pre-post relationship model, which is the focus of the statistical analysis of RD-D Trochim (2006). Thus, in principle the RD-D has comparable internal validity to randomised experiments, however, in practice, the internal validity measures are dependent on the accuracy of the modelling of the pre-post relationship of the data. In addition, RD-D requires a larger data set for statistical analysis comparable to randomised experimental design, Trochim (2006) states, as much as 2.75 times as many observations.

If the relationship between variables in a RD-D is not linear, the regression line will not represent the true relationship, resulting in an incorrectly fitted model. Common reasons in which this situation can arise is when the relationship is not linear between variables, but rather squared or cubic, or if the dataset is not normally distributed. Data transformation can be used to reduce bias, resulting from non-linear variables, allowing the correct modelling of the regression line. Another threat is the effect of the confounding variable which may interact at the classification cutoff point, and cause the treatment to be misinterpreted.

3.2.4 Statistical analysis

The purpose of analysing a hypothesis with statistical techniques is to determine if observed results are due to chance, termed the Null hypothesis. Alternatively, if the Null hypothesis can be rejected, it can be stated the treatment has some effect.

The statistical analysis undertaken is dependent on the experimental design, the aim of the hypothesis and the scale of the data explored. Descriptive statistics are used to describe the data presented, while inferential statistics are used to extrapolate the differences between groups of data into a generalisable condition. A range of statistical tests are summarised in Table 3.8 according to the analysis model, and experimental design.

Many statistical analysis techniques are reliant on the assumption of independence between variables, which is often not guaranteed in quasi experimental designs, due to the lack of random assignment of observations to control and treatment groups. Thus, quasi experimental analysis is more complex, and borrows from statistical tests commonly found in forecasting, such as Autoregressive Integrated Moving Average (ARIMA) .

Table 3.8: Appropriate statistical analysis models for identified experimental designs adapted from Shadish, Cook and Campbell (2002), Trochim (2006), Christensen, Johnson and Turner (2011), Montgomery (2013)

Experimental Design	Statistical Analysis model					
	T-Test	ANOVA	ANCOVA	Regression	ARIMA	Frequency
Randomised Two Group	✓	✓		✓		
Pre-test/Post-Test Control Group	✓	✓	✓			
Factorial Design		✓		✓		
Non-Equivalent Control Group			✓	✓		
Interrupted Time Series		✓		✓	✓	✓
Regression-Discontinuity			✓	✓		

3.2.5 Forecasting approach

Forecasting as an approach is traditionally used in economics and management disciplines to plan actions for future potential events, "predicted" from a historical based data model. Essentially, forecasting is the application of statistical methods on historical data with the aim of predicting future values with accompanying confidence levels. There are various methods employed in forecasting, which, similar to other scientific methods can be classified on a quantiative and qualitative continuum, where the polars are empirical experience (qualitative) and formal statistical based methods (quantitative), upon which specific methods are aligned. As determined from Section 3.1.4, the research is interested in quantitative methods, which can be further specified in the context of forecasting as either an Explanatory model or a Time series analysis model. All quantitative forecasting methods share three core assumptions, defined by Makridakis, Wheelwright and Hyndman (1998)

1. Data about the past is available
2. The Data can be quantified as numerical
3. Some aspect(s) of the past pattern will continue into the future (The assumption of continuity)

Additionally, Makridakis et al. (1998) notes that all forecasting methods follow a similar process, shown as Figure 3.3.

3.2.5.1 Exploratory analysis

The exploratory analysis of the data informs the class of forecasting methods which are appropriate, either explanatory or time series. Exploratory analysis involves the use of visual inspection of the data through the use of various types of plots, including time, seasonal and scatter, in addition to the application of descriptive statistics. The application of descriptive statistic techniques is dependent on the historical data type, however, no single test can definitively determine which forecasting technique to employ. Typically, the outcome of exploratory analysis is a shortlisted selection of models. As noted by Makridakis et al. (1998), forecasting models cannot account for all variations, or randomness in a model, meaning each forecasting model has an associated degree of error. The output of a system can thus be described as the functional relationship governing a system, and randomness, or $data = pattern + error$. Thus, at a high level, forecasting involves separating the pattern from the error, so that the pattern features can be used to define a model. The general process for estimating a pattern is the application, or fitting of a functional form (model) to minimise the error component of the equation, or, to eliminate randomness. Choosing and fitting a model involves enumerating through the shortlisted model selection, and fitting the dataset to each model, with the aim to minimise the error component of the pattern for accurate pattern development. Evaluation of each model is clearly dependent on the method selected, and the results of which can inform the acceptance of the model, further exploration or tuning of the model, or selection of a different model.

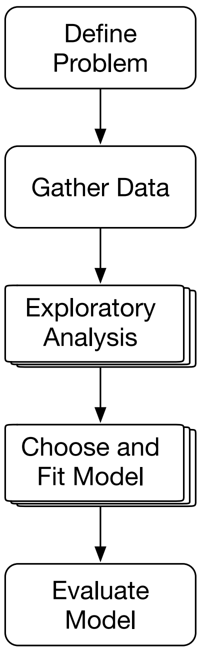


Figure 3.3: Summary of forecasting method processes, adapted from Makridakis, Wheelwright and Hyndman (1998)

3.2.6 Forecasting methods

3.2.6.1 Explanatory methods

Explanatory methods aim to describe and correlate the underlying relationships between variables in a data set, commonly, regression models are used to this effect. Explanatory methods can suffer in accuracy when used to forecast too far past the real dataset, with generalisability often an issue as explanatory methods describe correlation, and not necessarily causation.

3.2.6.2 Time series methods

Time series analysis is the statistical investigation of a temporally ordered series of observations from one or many cases. The purpose of the analysis can be to identify self correlated patterns offset in time, determine the impact of an intervention or experiment on an object of interest over time, or, in the case of forecasting, project the historically identified patterns forward for future predictions (Makridakis et al., 1998; Hyndman & Athanasopoulos, 2018). There can be multiple overlaid patterns in a data stream, which are identified as seasonality, trends and random error. Time series models are often an approach used when the independence of errors cannot be guaranteed, which counteracts a core assumption of regression models.

Time series methods take a black box approach to data analysis, where seasonal and trend patterns in temporally ordered data are identified, and then removed or transformed to forecast future data points. A class of commonly used time series models are ARIMA models, which use various combinations of auto regression, differentiation and moving average models.

3.2.6.3 Other methods

Additional methods exist which uphold the forecasting approaches core premise to predict future results on historical data, these are state-space models and artificial neural networks. Both the Kalman filter, a series of regression algorithms used for prediction (Welch & Bishop, 2006), and neural networks which rely on regression have been used previously as forecasting techniques (Makridakis et al., 1998).

3.2.7 Analysis of forecasting models

Measuring the accuracy of a forecasting method often means measuring the goodness of fit, or how well the model can reproduce known data. It is thus difficult to compare models directly as fitting measures vary between models, a comparison from a single criteria is of limited value. The comparison between models can be misconstrued, as many accuracy measures are reliant on the scale of the data, however relative error calculations can be used for comparison between models. In addition, the goodness of fit describes the models ability to identify historical data, but not the accuracy of the future forecasting result. However, the notion of accuracy can also be applied to the future forecast.

Comparison between different models is difficult, given the lack of a common measuring tool. Using a Naïve forecasting method provides a comparison tool for evaluating different models. Makridakis et al. (1998) details two naïve forecasts, named Naïve Forecast 1 (NF1), and Naïve Forecast 2 (NF2). NF1 takes the most recent observation as a forecast, which acts as a baseline for comparison. Any difference in the Mean absolute error and Mean absolute percentage error between the NF1 and other, more complex models can be discussed. NF2 considers the potential for seasonality in the data series, and thus removes seasonality from the dataset to create a seasonally adjusted data, which then follows the same forecasting method as NF1.

The accuracy measures presented however, treats each error as an equal, which is not the case given the potential scale differences between errors. A more robust method of comparison is Theil's U statistic defined in Equation 3.1 from Makridakis et al. (1998), which gives large error margins a larger weight than smaller error margins.

$$
\begin{aligned}
&U = \sqrt{\frac{\sum_{t=1}^{n-1}(FPE_{t+1} - APE_{t+1})^2}{\sum_{t=1}^{n-1}(APE_{t+1})^2}} \\
&\text{where} \\
&FPE_{t+1} = \frac{F_{t+1} - Y_t}{Y_t} \\
&APE_{t+1} = \frac{Y_{t+1} - Y_t}{Y_t}
\end{aligned}
\qquad \text{(Equation 3.1)}
$$

The Absolute Percentage Error (APE) defined in Equation 3.1 is the actual relative change, which is equivalent to the NF1, making the U statistic the comparison of the Mean Absolute Percentage Error (MAPE) of a given forecasting to the MAPE of NF1. Therefore, the accuracy of the forecasting method can be classified based on the following three points.

1. if $U = 1$ the forecasting model is as good as the naïve method
2. if $U < 1$ the forecasting model is better than the naïve method
3. if $U > 1$ the forecasting model is worse than the naïve method

Alternatively, using a similar approach to that of machine learning algorithms and experimental design, a test set and initialisation set can be used, whereby the initialisation set is used for estimating parameters and stating the method, while the accuracy of the measure is computed for errors in the test set only.

3.2.8 Simulation approach

Simulation can be a powerful tool in research which can improve the understanding of a phenomena of interest relating to an object or system (Dooley & Dooley, 2001; Rose, Spinks & Canhoto, 2015; Edgar & Manz, 2017). Complex models can be built at various levels of abstraction, which provides the ability to examine system interactions, component performance and test theories that are often not possible on real systems. Using a simulation model can be appropriate when the object of interest cannot be directly interacted with, due to a range of circumstances, such as cost of operation, criticality of the object of interest or associated risk of interacting with the system. This is particularly true for critical infrastructure (CI) systems, such as BAS and SCADA where building a real device test bed can be prohibitively expensive, and the risk posed by running tests on a CI is difficult to accept. Simulation methods are rooted in experimental design, whereby the aim of simulation is often to undertake specific experiments against a defined model, to optimise processes and provide alternative paths for future operation of a system. Simulations provide strong control over data generation and experimentation, however, results derived from a simulator are only as good as the underlying models used (Edgar & Manz, 2017).

For cyber security, Edgar and Manz (2017) defines simulation use as a computer process or application which imitates a cyber or physical process through generating similar responses and outputs. An abstract model of the actual process or object is built, instantiated into a program, and used to generate data which mimics the behaviour of the real system of interest.

Simulations can either be a black-box process, where the aim is to achieve realistic output from a system, or a white-box implementation, where the inner workings of the process are integral to the research questions. White-box implementations (termed emulation in Edgar and Manz (2017)) provide a higher fidelity simulation, using more complex models to provide high fidelity output compared with that of black-box simulations (Edgar & Manz, 2017).

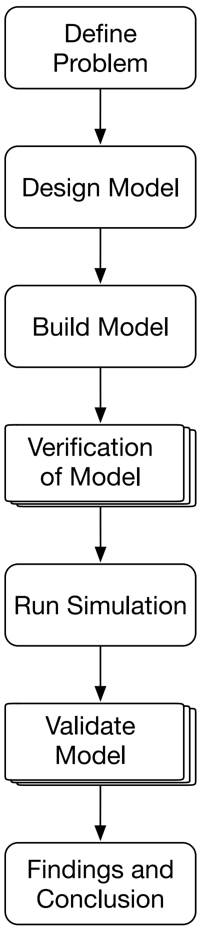


Figure 3.4: Summary of simulation method processes, adapted from Rose, Spinks and Canhoto (2015)

3.2.9 Simulation methods

Edgar and Manz (2017) state there are both general and cyber security specific methods for simulation. General simulation can be classified as agent-based, process-based (continuous), discrete, and Monte-Carlo methods (Dooley & Dooley, 2001; Edgar & Manz, 2017). Cyber security specific methods include network simulation, target simulation and threat simulation (Edgar & Manz, 2017). The purpose and potential use-cases for each stated simulation method are outlined in Table 3.9. Each simulation method follows a number of generic steps in order to undertake a simulation. A Flow Chart representation, adapted from Rose et al. (2015) for simulation methods is shown as Figure 3.4.

3.2.10 Simulation validation

Simulations are a specific implementation of reality based on an underlying model. To ascertain the validity of the simulation, the underlying models should be scrutinised. Typically, observational methods are used to test the validity of a simulation, by collecting like data of a system, and either

Table 3.9: Comparison of simulation methods, adapted from Dooley and Dooley (2001), Edgar and Manz (2017)

Class	Method	Purpose	Use-Case
General	Agent-Based	Define parts of a system as individual agents which operate independently from each other, data from interactions between agents and the defined environment are monitored	1. Discover emergent behaviour of the system 2. Represent distributed control of systems
General	Discrete-Event	A predefined, stochastic implementation which defines how and when variables change in the system	1. Event-based analysis 2. Interactions between variables not of interest
General	Process-Based	Generates behaviour based on a mathematical definition of the system	1. Simulate physical processes
General	Monte-Carlo	Execute a range of tests to determine likelihood of outcomes	1. Decision support based on likelihoods 2. Account for uncertainty from input parameters
Cyber	Network	Simulate network protocols to generate both normal and malicious network transactions	1. Protocol behaviour analysis 2. Traffic generation 3. Attack analysis
Cyber	Target	Controllable method of studying specific cyber incidents and vulnerabilities	1. Honeypots 2. Adversary profiling 3. Attack classification
Cyber	Threat	Analysis of user and system action when faced with a threat or hazard	1. Validation of experiments 2. Fault Analysis 3. Fuzzing 4. Attack classification 5. Exploit testing

comparing the data statistically to the simulation outputs, or using the real data as a baseline to generate models via machine learning methods. Using statistical methods, the distributions of behaviour can be derived, if the data fits distributions well, the distribution can be used to model and recreate similar behaviour in the simulation model (Edgar & Manz, 2017).

An alternative approach to validation is using experimentation, through defining tests cases which can be executed in the simulation, and the real system. A comparison between the test cases and real system responses can then be used to determine the validity of the simulation (Edgar & Manz, 2017). A downside of this approach is the requirement of direct interaction with the observed real process. Such as in the case of critical infrastructure and cyber-physical research, direct interaction to run test cases is not suitable.

3.3 Comparison of identified approaches

3.3.1 Candidate experimental designs

An experimental approach clearly fits to this research, determination of the specific design to use however, is dependent on the objective of the research, which is to identify known and unknown threats against BACnet/IP managed Building Automation Systems (BASs). The randomised two group design is not appropriate for this research, as a true experiment cannot be performed on the datasets generated. Specifically, samples provided to the algorithms cannot be randomly assigned, as an object of interest is the sequence and timing of network frames. Factorial designs can be used to compare combinations of independent variables. The independent variables defined in this research are the various algorithms used to test for anomalous network transactions. Given that differing data pre-processes are required for the application of each algorithm, an exhaustive comparison of multiple algorithms would not achieve a usable result, compared to direct comparison between specific algorithms using appropriate pre-processing techniques. Thus, factorial design is discounted. The research makes use of time series data, particularly data readings over time recorded from sensors, and network data with a key element of time, an interrupted time series design seems an appropriate fit for conducting experiments. However, the key use of ITSD is to identify if a treatment has had an effect, when the time of the effect is known. The stated assumption cannot be held when the aim of the research is to detect when a known or unknown attack occurs, rather than measure the effect of an attack on a system or network, thus ITSD is discounted. A regression discontinuity design would not be appropriate for this research, given the control set must be the real data for validity of the synthetic data, and the relationship between variables in the dataset cannot be claimed to be a linear relation, thus it is discounted. While the Non-Equivalent Group design can suffer from selection bias, additional controls discussed in Section 3.2.3.1, such as variable matching and statistical techniques can be used to account for selection bias. The Non-Equivalent Group design provides the strengths of a laboratory experiment, minus the random assignment, which fits the requirements of this research, thus is an appropriate design.

3.3.2 Candidate forecasting methods

A forecasting method can be used to answer a specific question about a future result, based on historical data and the assumption of continuity. Given the normal network interactions of a BAS are generally deterministic, there is the potential for a forecasting model to be used to define normality on the network, and identify relationships between features in the network. Subsequent models could be used to identify anomalous behaviour, based on the analysis of the forecasting method. Additionally, as time series models can account for seasonal changes, it is appropriate in a system where temperature variations and user interaction are based on seasons, to have a model which can take seasonality into account for identifying contextually normal and abnormal variable values. A range of other models are classified as forecasting methods by Makridakis et al. (1998), including state models and neural networks. Given the prevalence of state models in engineering and computer science research for learning, in addition to neural networks use in data analysis and cyber security, both methods are of interest. Specifically, Markov, Hidden Markov, and Artificial Neural Network methods.

The forecasting approach, by itself is not a complete answer for this research, as the use of forecasting based models is only a subset of tasks, rather than the overall objective of the research. Forecasting methods, however, fit well into an overarching experimental design, where specific methods are tested against a dataset as an experiment.

3.3.3 Candidate simulation methods

From analysis of the research goals there exists two distinct requirements for the simulation in this research. First, to provide a means of generating representative BACnet/IP network traffic in a controllable method based on a defined scenario. Second, to generate synthetic data values from physical phenomena, such as temperature, to proliferate over the simulated network. The outlined purposes in Table 3.9 informs the selection of a simulation method for both requirements. When discussing general simulation methods, Dooley and Dooley (2001) note that any simulation method can be applied to a problem with varying levels of success. In the context of this research, at a network level the system of interest closely resembles a discrete event system definition, whereby predetermined events (commands) occur in the system, changing variables and subsequently the state of the system in a stochastic manner. However, a discrete event simulation may be an oversimplification of an entire BAS, given there are also a number of non-constant variables which occur at the data generation level, specifically fluctuating internal sensor readings which cause commands to execute; therefore Discrete event is eliminated. An agent based simulation fits the requirements, given each device can have an associated agent model representing the distributed nature of a BAS, and generate data within each agent. Emergent behaviour between devices based on implemented communication rules would be appropriate for examining normal actions, and those of simulated attack traffic, and thus is selected. Additionally, a Process based simulation method could be used to generate the environmental data readings to be transmitted over the network, given they are continuous variables, many of which can be created using mathematical functions. There also exists uncertainty in the input parameters for some identified physical process to model, as such Monte-Carlo methods are also appropriate. The requirements of the research do not align with the Target method, as examining adversary interactions with BACnet systems is beyond the scope of this research, thus the Target method is eliminated. Similarly, the Threat simulation method is discounted, as the aim of the research is not to determine how devices and users act when a network attack is or has occurred, rather to perform analysis on the generated traffic. Naturally, the requirement to generate simulated network traffic of a specific protocol is within the brief of a Network Simulation, and thus is selected.

3.3.4 Selected design

The approaches identified through the use of Galliers' Taxonomy presented in Table 3.3 identified three possible comparative approaches for this research. Given the overlap between the derived approaches, due to a shared positivist paradigm, a range of methods and techniques are appropriate, and complementary for the research design. In consideration of the scope of the research, and the specificity of the forecasting and simulation approaches, it is not justifiable to select either as the research design of this research project. However, the usefulness of the methods which fall under these classifications cannot be dismissed. Both forecasting and simulation methods are rooted in experimental design, and as such can be used for specific experiments which output tangible results for the research. Of the examined experimental designs, the non-equivalent control group design was deemed most appropriate for this research, given the strengths of a laboratory experiment, with the flexibility of selecting specific groups for testing.

As such, a non equivalent group design is selected for use in this research, with the acknowledgement that some methods used can also be classified as methods used in forecasting and simulation approaches.

3.4 Research questions

Given the identified gap in knowledge, and the exploration of the philosophical underpinnings of the research, the following questions have been derived.

RQ_1 *How can known and unknown attacks against BACnet/IP based Building Automation Systems be detected?*

- SQ_1 *Are BACnet devices exposed to known threats?*
- SQ_2 *Do known BACnet attacks have distinguishable network patterns compared to normal BACnetwork traffic?*
- SQ_3 *Is machine learning applicable to identify known and unknown attacks against BACnet/IP networks?*
- SQ_4 *How accurate are machine learning approaches in detecting known and unknown attacks against BACnet/IP networks and devices?*

3.4.1 Hypotheses

The hypotheses derived from the research questions are defined as H_1 to H_5.

H_1 *BACnet devices are exposed to known threats*

H_2 *Known BACnet attacks have distinguishable network patterns*

H_3 *Machine learning is capable of identifying one or more known attacks against BACnet/IP networks*

H_4 *Machine learning is capable of identifying one or more unknown attacks against BACnet/IP networks*

H_5 *Hidden Markov Models are more accurate at detecting unknown BACnet/IP based attacks than known BACnet/IP based attacks*

3.5 Research design

The research design is defined by five distinct phases, visually represented as a framework in Figure 3.5. The five phases detailed in the framework include:

1. The *exploration phase*: which involved investigating the BACnet/IP protocol specification for understanding and identified potential vulnerabilities, retrieving known attacks from literature, examining existing simulation/network generation solutions, and collecting a range of BACnet device information from devices connected to the Internet.

2. The *contextualisation phase*: where analysis of collected device information and threat modelling of defined BACnet devices and networks were undertaken. Further, potential algorithms and techniques for anomaly detection were defined and selected for the research. Finally, a trial dataset was generated to test existing simulators, implement a known attack and test the shortlisted algorithms.

3. The *data collection phase*: which was split into two stages. Stage one involved the collection of real data. Stage two detailed the generation of a synthetic dataset, which involved defining a scenario, devices to simulate, and data generation algorithms of both normal and attack type. Stage two also involved simulation validity testing.

4. The *experimental phase*: which involved preprocessing the datasets for the appropriate algorithms, application of the selected algorithms to the generated and real datasets, and recording the results.

5. Finally, the *analysis phase*, examined the results of the experiments to suggest answers to the posed research questions and hypotheses.

3.5.1 Phase One: Exploration

The aim of phase one was to explore how the BACnet/IP protocol operates, how many devices are directly accessible over the Internet, and how to generate BACnet/IP network communications. BACnet has five different major standard revisions, with multiple addenda. At the time of this research, the ASHRARE-135-2012 BACnet standard was the most recent, which incorporated both BACnet/IP and the BACnet Security Services (BSS) features. As such, the research focused on the 2012 version of the standard, with acknowledgement that other versions are also used in real settings. The protocol specification was examined, with the data transmission and command sections targeted for further analysis in regards to the security principles of confidentiality, integrity and authentication. Further, identified from the literature review, a list of published BACnet specific attack code was detailed, examining which commands were typically used to undertake attacks against BACnet devices. In order for the research to proceed, a range of BACnet simulation options were identified and examined for use in the research.

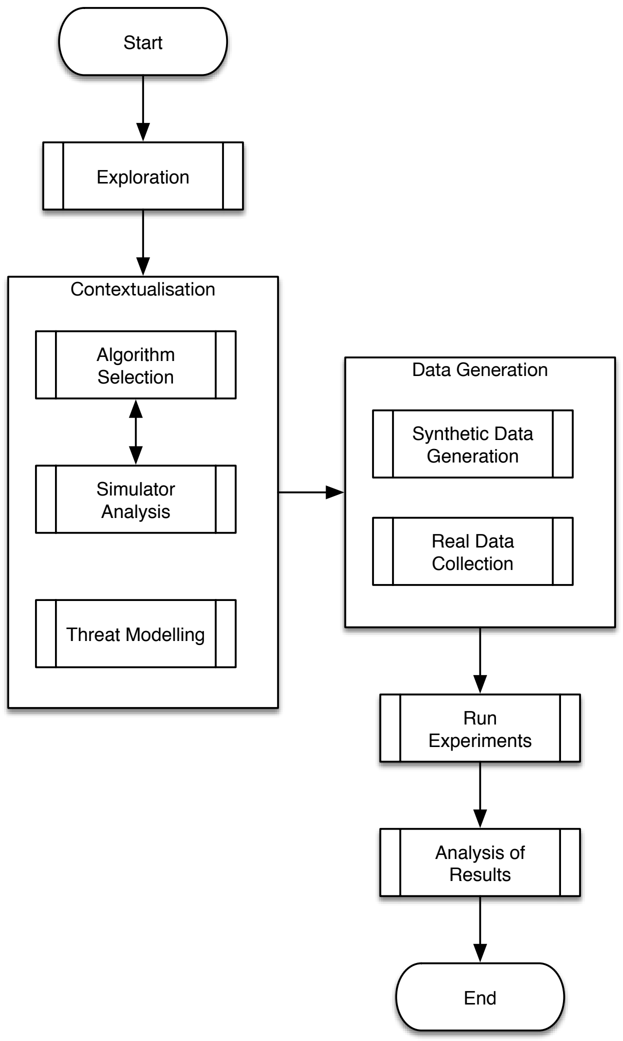


Figure 3.5: The framework representation of the research design undertaken

To explore the connectedness of BACnet devices to the Internet, a means of collecting device information from BACnet devices was investigated. Censys (Censys, 2018), an organisation which frequently scans the Internet for specific protocols had the required open-access data for this research. The data retrieved was stored for further analysis.

3.5.2 Phase Two: Contextualisation

The contextualisation phase consisted of four sub-phases. First, to undertake analysis on the retrieved BACnet device data, to identify a range of details, including location, device profiles, device vendors and common configuration settings. This analysis fed forward into the device profiles selected in Phase three. Second, contextualising the threat to BACnet devices and networks was defined using a threat modelling approach. The STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege) threat taxonomy (Howard & Lipner,

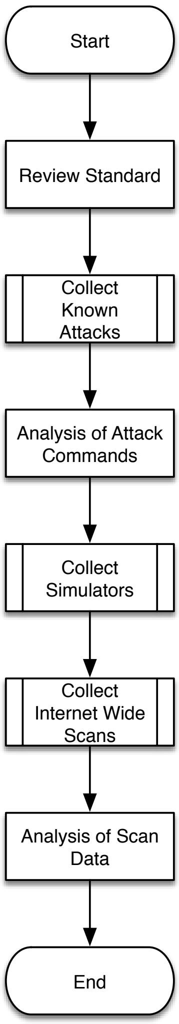


Figure 3.6: Flow chart for Phase One of the research

2006) was selected given its prevalence in security research, further, modelling was taken from a device profile and network level. Third, narrowing the scope of the research by shortlisting a range of machine learning algorithms, and testing them against trial datasets. Fourth, generating a trial dataset to test existing BACnet simulation software, implement a known BACnet attack, and trial run a range of algorithms defined in the selection sub-phase. Where an algorithm or process did not perform according to expectations, the selection process would iterate.

3.5.3 Phase Three: Data collection

The data collection phase was split into two sub-phases, real data collection and synthetic data generation. To test attacks against a real BACnetwork is not feasible, as such a simulation was required. To validate the simulation, real data was collected for statistical comparisons, network topology definitions and example scenario definitions. The results from Phase two, sub-phase four regarding simulation testing fed into the definition of the simulation. Particularly, the requirement to generate underlying synthetic data to traverse the network using the BACnet/IP protocol. Data generation algorithms were defined, for both normal traffic and attack traffic. A simulation environment was developed, which operated at a device and network level for generating data based on a given scenario, with intermittent BACnet specific network attacks undertaken. The validity of the scenario components was undertaken in this phase.

3.5.4 Phase Four: Experimental

The Experimental phase consisted of taking both the real-normal, synthetic-normal and synthetic-attack datasets for testing of selected algorithms. Preprocessing was a core process of this phase, to ensure the efficacy of the application of each algorithm.

3.5.5 Phase Five: Analysis

During the analysis phase, the results from each algorithm-dataset pairing was analysed, and statistically compared. A range of measures relating to binary classification were used to evaluate the algorithms, namely, the false positive rate, true positive rate, false negative rate, true negative rate, and Matthews correlation coefficient. The precision and accuracy of each algorithm were also evaluated.

3.6 Variables

There are a range of variables defined in the research of varying types.

3.6.1 Dependent variables

The dependent variables for the research were identified as the features which the selected algorithms were applied to. These include the Δt between two successive network frames, the BACnet command which was executed and the packet size,

3.6.2 Independent variables

The independent variables for the research were defined as the specific algorithms and methods used in the research.

1. Hidden Markov Model
2. Artificial Neural Network
3. Graph analysis
4. k-means clustering

5. Gaussian mixture model clustering
6. Time series analysis

3.6.3 Control variables

To ensure the internal validity of each algorithms run, a number of control variables were defined. The materials used in the data collection and analysis phases were held identical for each independent variable, see §3.7. Datasets generated in the contextualisation phase were undertaken earlier than the data generation phase. As the network stack used was under active development, various versions of the network stack were used. Each dataset generated used a specific version of the selected network stack, defined in §3.7 As each dataset is different, comparisons between independent variables are compared on a per dataset basis.

3.6.4 Compound variables

The pseudo random number generation functions of *Python* were used in a number of the synthetic data generation algorithms. A common seed value was selected for the *Python random.seed* function, allowing for the repeatability of data generation to occur.

3.7 Materials

A range of software and hardware were used throughout the research, detailed in Table 3.10. As the research progressed, the environment used to generate datasets was iterated upon and improved, as such, the materials used are classified by research usage.

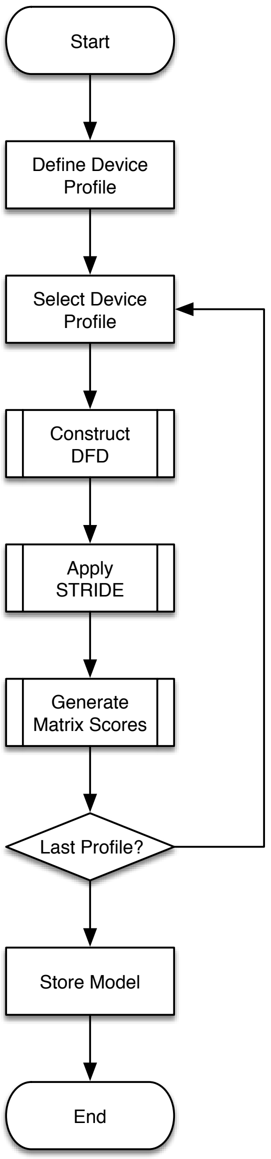


Figure 3.7: Flow chart for undertaking threat modelling of BACnet device profiles for Phase Two, Sub-phase Two of the research

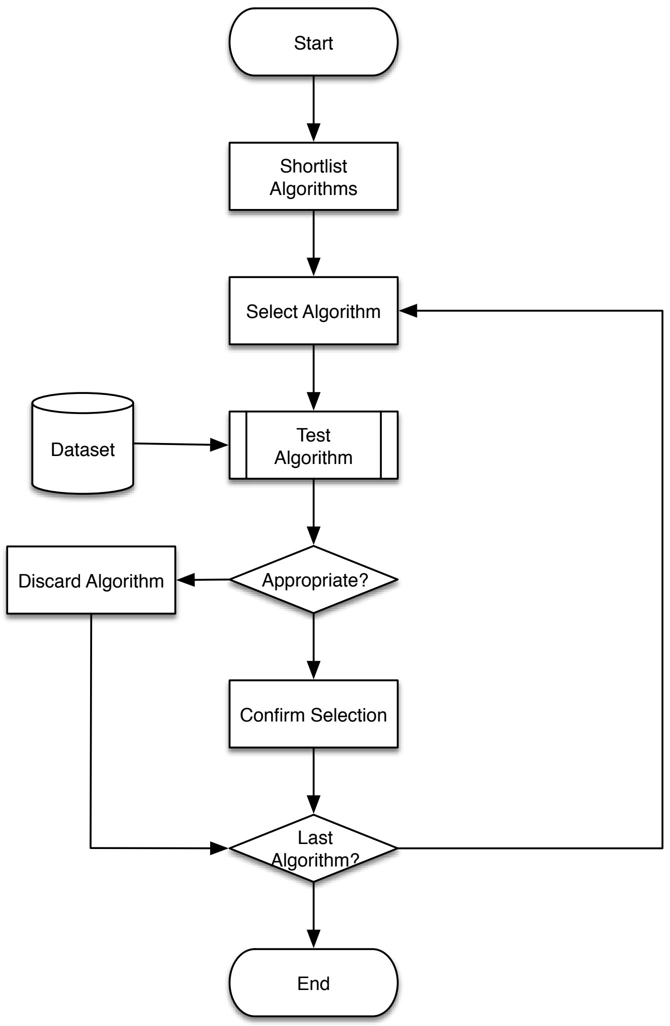


Figure 3.8: Flow chart of Algorithm selection and Testing for Phase Two, Sub-Phase Three of the research

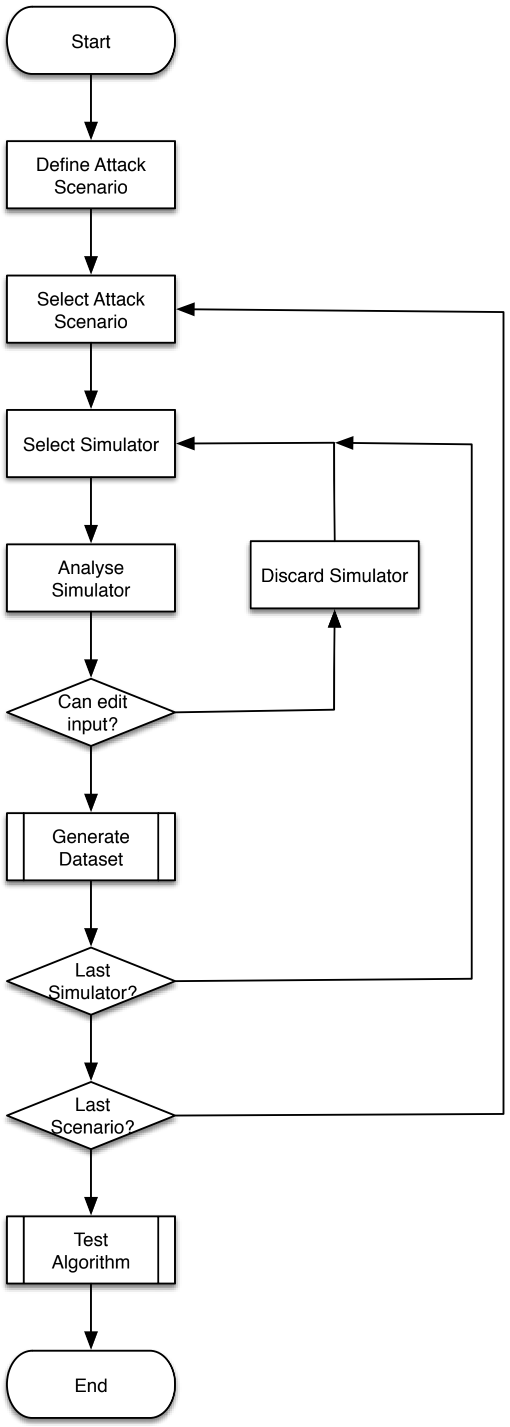


Figure 3.9: Flow chart for analysis for preliminary testing of simulators, and algorithms Phase Two, Sub-Phase Four of the research

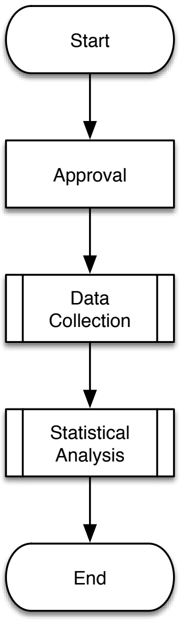


Figure 3.10: Flow chart for Phase Three, Sub-Phase One of the research

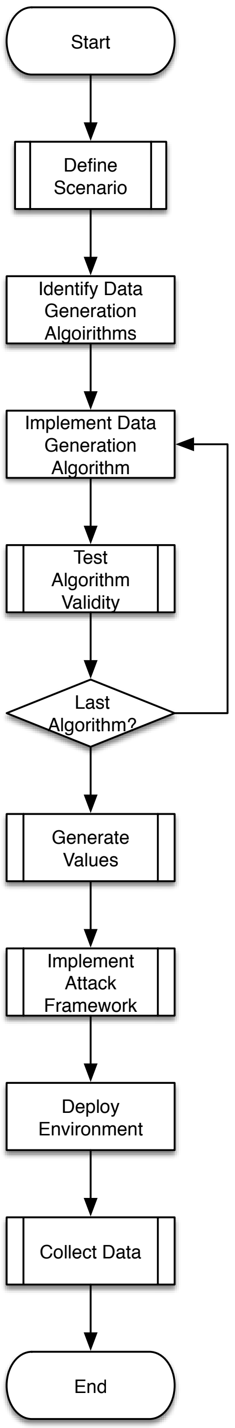


Figure 3.11: Flow chart for Phase Three, Sub-Phase Two of the research

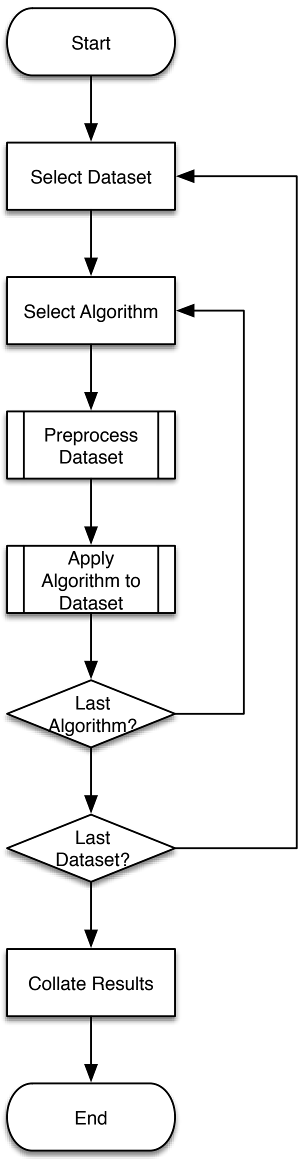


Figure 3.12: Flow chart for Phase Four of the research

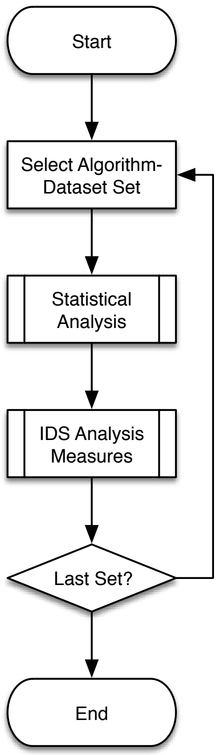


Figure 3.13: Flow chart for Phase Five of the research

Table 3.10: Materials used throughout the research, classified by usage

Research Usage	Hardware/Software	Details	Description
Development, Analysis and Writing	Macbook Pro	2015, 16GB ram, 2.9GHz i5 processor	-
	OS X	10.11.6	OS
	Jupyter Notebook	1.0	Development Environment
	Jupyter Core	4.4.0	Development Environment
	matplotlib	1.3.1	Graphics
	ipython	5.4.1	Development Environment
	python	2.7	Programming Language
	numpy	1.8.0rc1	Python Math Package
	pandas	0.20.2	Python Data Science Package
	pomegranate	0.8.1	Python Markov Model Package
	Bibdesk	1.6.11	Referencing
	TexPad	1.7.9	Word Processing
	Omnigraffle	6.2.3	Graphics
	Wireshark	2.2.6	Network Analysis for Mac
Theory Testing Dataset	Raspbian Jessie	R2016-09-23, V4.4	Rpi OS
	CBMS Studio	V1.3.7.1204	Simulation Software
	CBMS Engineering configuration tool	V1.3.1221.7	Simulation Software
	Bacnet Open Stack	V0.82	Networking Stack
	Windows 7	SP1	OS
	Windows 7 Desktop	16GB Ram, AMD FX-8120 eight core processor	Desktop
	Wireshark	V1.12.1	Network Analysis
	3x Raspberry Pi 2	1GB ram, 16GB SD card	Sensors
	Cisco Switch	Catalyst 3560, SPAN configured	Networking

Research Usage	Hardware/Software	Details	Description
Trial Dataset	SCADA Engine BACnet device simulator	2.0	Simulation Software
	Windows XP	SP2	Virtual Machine OS
	Windows XP VM	512 Mb ram, 1 processor core	Virtual Machine
	Ubuntu	13.04, 64bit	Virtual Machine
	Ubuntu VM	4GB ram, 1 processor core	
	Bacnet Open Stack	V0.82	Networking Stack
	Macbook Pro	2015, 16GB ram, 2.9GHz i5 processor	-
	OS X	10.10.5	OS
Real Data Collection	Ubuntu	16.04 LTS, 64bit	OS
	Ubuntu Desktop	16GB Ram, AMD FX-8120 eight core processor	Capture Machine
	Wireshark	2.2.6	Network Analysis
	Dumpcap	2.2.6	Command Line Wireshark
Simulation Dataset	Vmware Fusion	6.0.6	Virtual Machine hosting
	Vmware Fusion	8.5.0	Virtual Machine hosting
	Windows 7	SP1	OS
	Windows 7 Desktop	16GB Ram, AMD FX-8120 eight core processor 64bit	Desktop
	Cisco Switch	Catalyst 2960, SPAN configured	Networking
	ESXi Vmware	6.0	Virtual Machine hosting
	ESXi Vmware Server	32GB ram, AMD FX-8120 eight core processor	Virtual Machine hosting
	Lubuntu	16.04	Virtual Machine
	Actuator VM	256 MB Ram, 1 CPU core	Virtual Machine
	Sensor VM	256 MB Ram, 1 CPU core	Virtual Machine

Research Usage	Hardware/Software	Details	Description
	Thermostat VM	768 MB Ram, 1 CPU core	Virtual Machine
	VAV box VM	256 MB Ram, 1 CPU core	Virtual Machine
	Workstation VM	768 MB Ram, 1 CPU core	Virtual Machine
	Logger VM	256 MB Ram, 1 CPU core	Virtual Machine
	Capture VM	768 MB Ram, 1 CPU core	Virtual Machine
	BACnet Open Stack	V0.85	Networking Stack
	BACnet Open Stack	V0.85-a	Networking Stack amended
	Ubuntu	16.04 LTS, 64bit	Virtual Machine
Flow Analysis	Ubuntu VM	16GB Ram, 4 processor core	Virtual Machine
	YAF	2.0.0	Flow Generation
	Moloch	0.19.2	Network Analysis
	Gephi	0.9.2	Graph Analysis and Community Detection

3.8 Limitations

The research conducted focused on BACnet/IP traffic which travels over IP based networks. While some serial traffic is encapsulated in IP, the investigation did not specifically look at monitoring, nor simulating the serial-based media of BACnet managed BAS. Serial-based communication medium are traditionally used for the connections between sensors and controllers to reduce costs of system implementation (Cisco, 2008). However, the BACnet standard is independent of the physical layer implementation of the network, given the aim of the research is to assess the BACnet/IP version of the protocol, sensor devices are implemented with BACnet/IP as the communication medium. The representation of a device in BACnet is abstracted away from the communication medium, thus allowing for the definition of BACnet/IP based sensors.

There is no assurance that the real dataset collected and used in the investigation does not contain malicious traffic. Given that real data is used to validate the simulation, a necessary assumption is made that there is no anomalous behaviour in the collected data.

3.9 Threats to research

Of the defined threats in Table 3.7, History, Testing, Instrumentation, Regression, Selection and Selection Interactions can be seen as threats to this research. While Maturation and Mortality are not applicable threats to this research due to the lack of human element. Table 3.11 details the mitigations applied to combating the identified research threats, through the design and application of the experiments.

Table 3.11: Potential mitigations to the associated threats of using experiments

Threat	Employed Mitigation
History	Isolated simulation network to prevent external interference
Testing	Separate training and testing data sets
Instrumentation	Consistent simulation, working environment and materials
Regression	Identical training and testing datasets for all algorithms. Performance criteria stays constant through testing and training phases
Selection	Identical testing dataset allocation for testing of each algorithm
Selection Interactions	Identical training and testing datasets for all algorithms, Performance criteria stays constant through testing and training phases

3.10 Validity

Simulation validity was discussed in Section 3.2.10. Validation of each model used in the simulation is undertaken. Where possible, the physics-model of the phenomena is used based on accepted laws relating to the nature of the universe. The selected models are typically used in the design and analysis of HVAC systems and Building Energy Simulation programs, discussed further in §4.6. Thus deemed appropriate for data generation for this simulation. Where accessible, the data generated in the simulator was compared to real data, collected from external weather files. Further discussion on validity is directed to the specific algorithm implementation sections in Chapter 4. The Network topology was designed to represent a typical HVAC managed BAS, with topologies taken from vendor documentation, discussion with BAS experts, and descriptions from the BACnet standard, further discussed in Chapter 4. Device selection and definitions are based on national and international industry standards, and configuration retrieved from vendor information, and Internet scannable device configurations.

Chapter 4

Exploratory Results

This chapter describes a range of initial results generated as part of this research. In §4.1, the results derived from investigating the BACnet/IP protocol are presented, with two commands noted for further investigation. §4.2 and §4.3 outline the existing threat to BACnet devices, presenting collated data from Internet wide network scans and threat modelling of a network controller. Next, one of the identified commands is modelled and tested using a network stack to determine the viability of a theoretical vulnerability. In §4.4, a number of existing BACnet/IP simulators are examined, with datasets generated to test the capability of the simulators generating normal and malicious traffic. Two datasets were generated, the first to examine the use of a machine learning algorithm for identifying a frequency attack, and the second to investigate the remaining command identified in §4.1. Penultimately, §4.6 details the real network data collection, followed by the design and implementation of a BACnet/IP network testbed. Finally, an attack framework is presented which was applied to the defined network testbed for attack data generation.

4.1 Protocol analysis

This section describes a preliminary investigation of the BACnet protocol specification. The purpose of this task was to identify additional objects and services which may be used to perform legitimate-yet-malicious, commands. The section focusses on two identified components, namely, *Change of Value* reporting functions, and the queuing implementation for priority commands. An overview of the limitations of the security addendum is outlined in §2.5, and is thus not repeated here.

4.1.1 BACnet change of value reporting

BACnet systems are data driven, with values passed over the network between devices using various methods. As BACnet is a peer-based network, any capable device in a BACnetwork may request values from other devices, or be notified of events occurring. To accommodate peer-based communications as default, BACnet devices are passive servers which listen for requests and service received requests. Each data sharing transaction is represented as a client/server request, where the device requesting data is a client, and the targeted device providing the data is the server. A server in this case might be a sensor or actuator, while a client might be a controller or workstation. Data can be shared using one, or multiple methods simultaneously. There are three defined methods, polling, triggered collection and *Change of Value* (CoV) reporting.

Polling is a simple method, where data requests are made at pre-defined time intervals. Polling has the potential to miss value changes if they occur between time intervals. As noted in Chipkin (2009), a balance is required in regards to the polling interval, too large and data will be missed, too small and the network will be impacted by the traffic size. Triggered collection defines a boolean property in a device, when the property is true, data is retrieved from the device. External network writes and internal processes such as alarms or other events can cause the trigger to become true, and cause an immediate acquisition of values to occur. *CoV* reporting is an active data collection method, defined in SSPC-135 (2012, pp. 461-464). *CoV* reporting defines subscriptions between devices, where a threshold value is set. If the monitored value changes over the set threshold, a notification is sent to all subscribed devices. The server device maintains a list of subscribers, namely the *Active_CoV_subscriptions* which holds the CoV subscriptions. CoV subscriptions may be either *Confirmed* or *Unconfirmed*, reminiscent of TCP and UDP in operation. *Unconfirmed* CoV reporting sends a notification to the subscriber when a value changes within the CoV threshold of the subscription. *Confirmed* CoV reporting incorporates acknowledgement of change, with an ACK packet to be sent from the subscriber to the device serving the data.

Due to the variety of media on which BACnet operates some devices take longer to acknowledge a *Confirmed* CoV notification than others. Two device object properties, *APDU_timeout* and *Number_Of_APDU_Retries*, set on the client device determine how long to wait for an acknowledgement, and how many times to retry waiting. The values for these properties are vendor and even device-specific. The BACnet standard suggests between 6,000ms and 10,000ms (Newman, 2013); the de-facto standard set by the vendors is a 3,000ms wait time with three retries (see Table 4.1). However, some vendor guidelines suggest a 20,000ms or even 60,000ms wait time, dependent on the capability of older devices. Further, one guide suggests all *APDU_timeouts* should be set to the highest value in the system (Contemporary Controls, 2014). In contrast, Siemens (2012) states that any timeout over 30 seconds is too long. If a subscriber is offline when a *Confirmed* CoV notification is sent, the CoV server device will wait the length of the *APDU_Timeout* of the client device, and then retry the CoV notification the specified number of times before processing the next CoV notification. Given the length of some *APDU_timeout* values and the number of retries, network delays can occur (Chipkin, 2009).

Additionally, when a subscriber goes offline, CoV messages are not stored or queued, therefore if a subscriber returns to the network, data synchronisation can be lost (Chipkin, 2009). If the subscription is *Unconfirmed*, there is no feasible way to determine if the subscriber has received the CoV notification, or tell if the subscribing device is offline. A combination of polling and CoV is suggested to counteract devices power cycling, however the solution is not complete, as the logging device for the system could suffer from the same issue, and the log of a CoV will never be recorded (Chipkin, 2009). Subscriptions to devices are not persistent between power cycles, meaning if a device is reset for any reason, the subscriptions to other devices will not be preserved and must be re-connected. Automatic re-subscriptions can be implemented, where a duration property exists which triggers a CoV subscription to occur. However, automatic re-subscription is also not persistent between power cycles.

Table 4.1: Sample of BACnet vendor APDU-Timeout and retry default values and ranges, expanded from Peacock, Johnstone and Valli (2018, p259)

Vendor	Default APDU_timeout value (ms)	Default APDU retries	APDU_timeout value (ms) Range	APDU retries Range
ScadaEngine[1]	500	5	300-30,000	0-5
Kepware[2]	1,000	3	100-9,999	1-10
Siemens[3]	3,000	3	-	1<
Contemporary Controls[4]	3,000	3	-	-
Tridium[5]	3,000	3	-	-
UTC[6]	6,000/10,000	3	-	-
Obvius[7]	6,000	3	6,000<	-
Metasys[8]	20,000	3	500-20,000	-
Viking Controls[9]	3,000/60,000	3	-	-

[1] (Scada Engine, 2009b)
[2] (Kepware, 2016)
[3] (Siemens, 2012)
[4] (Contemporary Controls, 2014)
[5] (Tridium, 2017)
[6] (UTC Fire and Security, 2015)
[7] (Shepard, 2013)
[8] (The S4 Group, 2015)
[9] (Viking Controls, 2002)

Due to the limited capacity of BACnetworks oversubscription of CoV notifications is plausible. Robust testing of oversubscription is often not carried out due to the risk of damage to devices (Chipkin, 2009; Newman, 2013). Further, many devices have a maximum subscribers limit, which is often short according to Chipkin (2009). There is no maximum limit of subscriptions defined in the BACnet 2012 revision 19 of the standard (SSPC-135, 2012, p461). A device on the BACnetwork may subscribe to the same object multiple times, as the unique identifier for each subscription is self-assigned. Each implementation of a device may have a limit applied to the quantity of subscriptions that each device can initiate. This bottleneck can create a network security issue, where critical devices will not receive notifications due to the subscription limit being reached (malicious or not) (Peacock et al., 2018).

A potential scenario defined in Peacock et al. (2018) is discussed here. A malicious device could send *Confirmed*, low threshold value CoV subscriptions to every supported device on the BACnetwork, and then disconnect from the network. Whenever a value on any device changes, the malicious device will be notified, but as the malicious device is offline, each legitimate device will

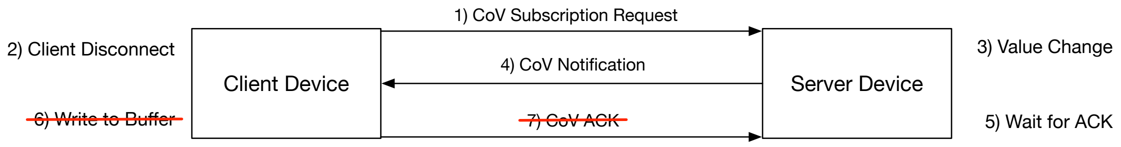


Figure 4.1: Malicious confirmed *CoV* transaction, red struck-out labels indicate normal *CoV* transaction steps which do not occur due to the attack, replicated from Peacock, Johnstone and Valli (2018, p261)

then wait for the timeout to expire before sending the next notification. As the *APDU_timeout* property is defined on the client of the transaction, the malicious device may set the length to wait, and the number of retry attempts. A model of the attack is detailed in Figure 4.1, and explored in §4.4.4 and §5.4.

4.1.2 BACnet bounded priority arrays

In a similar fashion to reading data, any capable device in a BACnetwork may write to any writable property on any other device. As some property values directly cause cyber-physical actions to occur, conflict resolution in the form of a priority system is implemented. BACnet accounts for the potential of conflicting commands through a conflict resolution process where properties are split into commandable, and writeable types. *Commandable properties* are defined as those whose value change causes physical actions, while all other properties are defined as writeable. Conflict resolution is only applied to *commandable properties*, whereas writable properties have no priority mechanism, meaning the last write to the property overwrites the previous value.

Priority arrays are defined in Section 19.2 of the BACnet 2012 standard. The *present value* property of many objects in BACnet are classed as *commandable properties*, for example, *Analog Output* and *Binary Output* objects. BACnet devices interact with *commandable properties* using the *Write property* or *Write property multiple* service requests. The request primitive for both services contain three parameters; Property Identifier, Property Value and Priority. The three parameters contain the *commandable properties* ID, the desired value for the property, and the priority value respectively. The priority value is a number set between 1 and 16 for that *Write property* service request, the lowest number having the highest priority.

Outlined in Peacock et al. (2018), the BACnet standard defines consistent representations for the applications of command priority levels, with five defined applications and eleven flexible open applications which can be implementation specific, outlined in Table 4.2. When a client device no longer requires access to the commandable property in a service provider, a relinquish command is sent to the provider, using *Write property* or *Write property multiple* service requests. The relinquish request uses the same parameters as a normal write request with the property value parameter set to *NULL*. When a device is notified that no service request exists at that priority level, the next priority array element is checked for a service request. When all elements in the priority array are *NULL* the property value will be set to a default value, defined in the *Relinquish_Default*

Table 4.2: BACnet priority array applications, replicated from Peacock, Johnstone and Valli (2018, p257)

Priority level	Application
1	Manual_Life Safety
2	Automatic_Life Safety
3	Available
4	Available
5	Critical equipment control
6	Minimum on/off
7	Available
8	Manual Operator
9	Available
10	Available
11	Available
12	Available
13	Available
14	Available
15	Available
16	Available

property of the object.

Each priority value in the array may only hold one command value at a time. If two devices have written to a commandable property with the same priority level it is unclear which value has precedent. As source authentication is not specified for write commands entering the priority table, the priority array attempts to create a queue of values to enter into a commandable property. This results in a similar scenario to writable properties, where last-write-wins occurs. Seemingly, this negates the purpose of the priority array implementation. Further, upon relinquishing an array position with a *NULL* value to the array position, the standard describes "unknown behaviour" for any queued command which is in the same array position (SSPC-135, 2012). Due to a lack of verification or limitation on which devices may issue specific priority levels, any capable device may change the value of a commandable property at any priority (Peacock et al., 2018). As these actions are defined normal in the standard, it is difficult to detect malicious use of this service. Detection approaches to write commands are outlined in Johnstone et al. (2015), and expanded on in §4.4.1. Further exploration of the priority array behaviour is undertaken in §4.3.1, and §5.4.

4.1.3 Summary of protocol analysis

Analysis of the protocol specification identified two potential issues which could manifest as vulnerabilities. Both required further investigation, as it was unclear from the specification what would be the correct behaviour of devices if the identified scenarios were to occur. Further exploration is undertaken in §4.4.4 and §5.4 for the *CoV* reporting function, and §4.3.1, and §5.4 for the priority array, respectively.

4.2 Scan analysis

A number of authors have discussed the exposure of BACnet devices at various points in time, identified in §2.4. As a longitudinal view of exposure of BACnet devices, Internet wide scans from Censys (Censys, 2018), an open-access repository for researchers were retrieved and analysed. Scans from the period December 2015 to January 2018 were retrieved. Over the period, 102 scans of the Internet were undertaken against the BACnet default port 47808. 1,733,392 devices were probed, with 76,489 unique devices identified. Over the entire duration, 3,670 devices were active in every scan. See Figure 4.2 for a general perception of global location of unique devices over the scan duration, derived from the *Geolite 2* IP database (MaxMind, 2018). Further, Table 4.3 outlines the ten countries with the highest proportion of scanned BACnet devices. Of note, Australia is ranked 5th overall, with 3.585% of devices over the three year duration. In a similar outline to Gasser et al. (2017), Table 4.4 outlines the five vendors with the highest representation of unique devices in the Censys scans.

Table 4.3: Breakdown of countries and the number of unique hosts scanned from December 2015 to January 2018

Country	Unique Scanned Hosts	Percentage of Total Unique Hosts
United States	30942	40.453%
Canada	9618	12.574%
France	4260	5.569%
Germany	2994	3.914%
Australia	2742	3.585%
Oman	1787	2.336%
Spain	1725	2.255%
Brazil	1553	2.030%
Italy	1536	2.008%
United Kingdom	1504	1.966%

Table 4.4: Unique device vendor counts identified over the duration of the examined scans, 2015-2018

Vendor	Unique Device Count	Percentage of Scanned Devices
Delta Controls	7661	10.0%
Reliable Controls Corporation	7501	9.8%
Tridium	4953	6.5%
Automated Logic Corporation	3456	4.5%
SAUTER	2094	2.7%

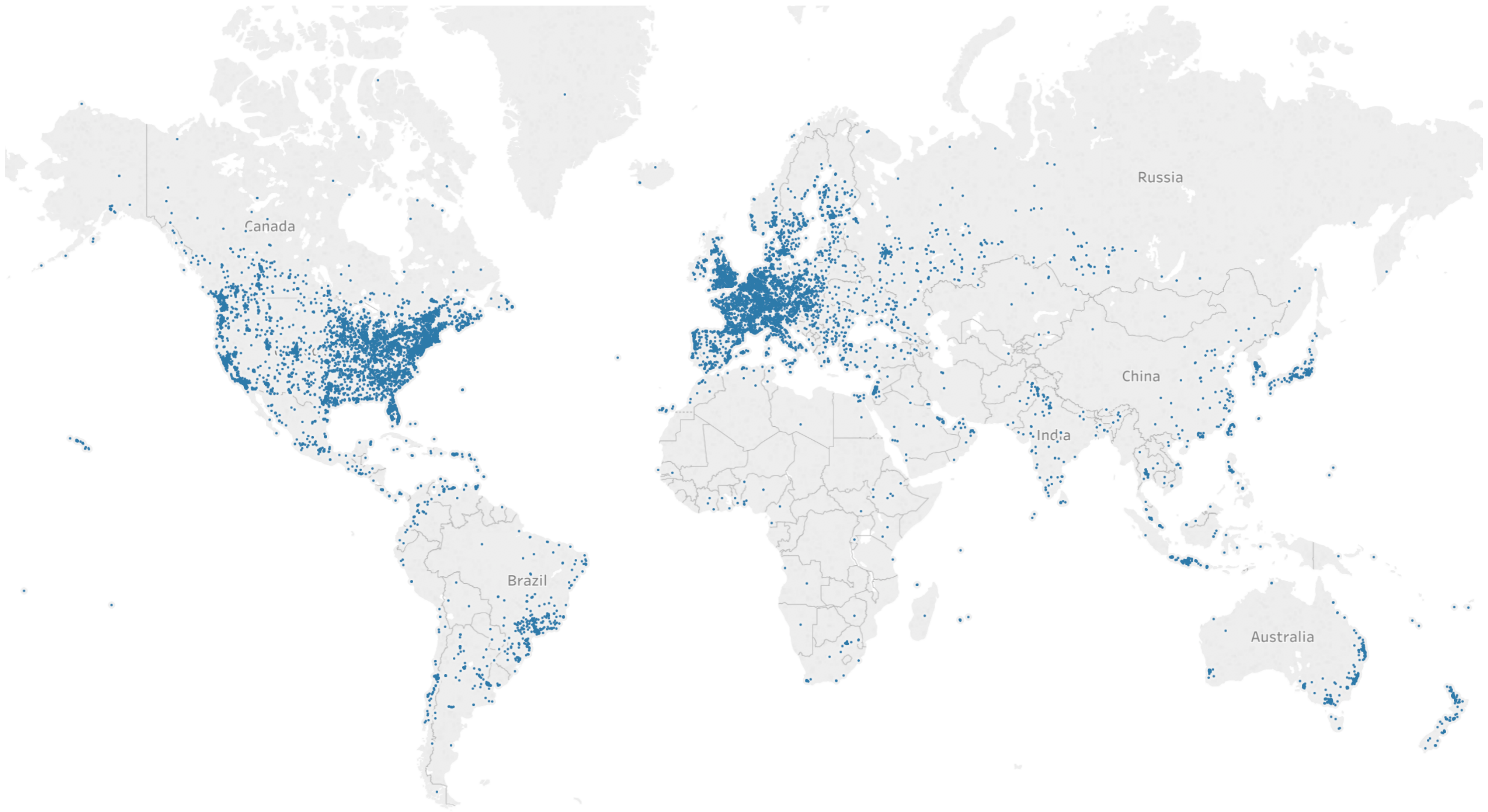


Figure 4.2: Overview of global unique device locations derived from Censys scans and Geolite2 geolocation database

4.3 Threat modelling

Threat models are often used to identify threats and systematically reveal vulnerabilities in a system (Peacock et al., 2017, 2018). The core requirements for a threat model are the identification of adversaries and their motivations, description of the system and identification of threats against the system. The last published threat model undertaken for BACnet was by Holmberg (2003). Adversaries, motivations and the connectivity of BAS have changed in the past 15 years, as such threat modelling was undertaken to ground the research perspective. There exist many taxonomies regarding adversary identification and classification, such as those presented in Magar (2016). Further, generic adversaries and their motivations (see Table 4.5) were identified in Bernier (2013), and presented in Peacock et al. (2018).

Table 4.5: Summary of cyber adversaries, adapted from Bernier (2013), replicated from Peacock, Johnstone and Valli (2018, p266)

Adversary	Modus Operandi	Motivation
Novice	Denial of Service Pre-written Scripts	Attention Seeking Prestige
Hacktivist	Denial of Service Defacement Information Disclosure	Political Cause
Insider	Sabotage Information Disclosure Internal Knowledge	Revenge
Coder	Develop scripts	Power Prestige
Organised Crime (Blackhat)	Avoid Exposure Highly Skilled Well Resourced Targeted Attacks	Money Greed
Cyber Terrorist	Destabilise Cyber or Physical assets Disrupt Cyber or Physical assets Destroy Cyber or Physical assets Highly Skilled Well Resourced	Ideology
Nation-State	State Sanctioned Destabilise Cyber or Physical assets Disrupt Cyber or Physical assets Destroy Cyber or Physical assets Highly Skilled Well Resourced Information Theft	National Interests

As noted by Newman (2013), the threat considered most damaging originally to BACnet was the insider adversary, given the previously segregated nature of BAS networks, and the system knowledge required. With increased connectivity and highly trusting devices, adversaries now have external pathways to interact with BAS directly, or use BAS to pivot into enterprise networks (Peacock et al., 2018). Given the defined *modus operandi* in Table 4.5, the motivation of each adversary could be fulfilled when targeting an appropriate class of building system, e.g hospital, office complex, government building.

Building on the threat modelling undertaken in Holmberg (2003), known BACnet specific vulnerabilities were collected from a range of sources (Holmberg, 2003; Kaur et al., 2015; Johnstone et al., 2015; Caselli, 2016; Tonejc et al., 2016; Esquivel-Vargas et al., 2017; Peacock et al., 2017) and summarised in (Peacock et al., 2018). The specific commands and properties used in these vulnerabilities were identified, and classified using a STRIDE threat matrix (Howard & Lipner, 2006), presented as Table 4.6. To quantify the defined known attacks to a scenario, a model of the interaction between a controller and other devices was defined in Figure 4.3. The model details commands, objects and properties which are interacted with, sent or received by the controller device.

Table 4.6: Known attacks classified against the STRIDE matrix, replicated from Peacock, Johnstone and Valli (2018, p268)

Attack Type	Specific Command/Property	S	T	R	I	D	E
Denial of Service	*Subscribe-CoV*	✓	✓			✓	
Denial of Service	*Router-Busy-To-Network*	✓	✓			✓	
Denial of Service	*I-am-Router-to-Network*	✓	✓			✓	
Denial of Service	*Router-Available-to-Network*	✓	✓			✓	
Denial of Service	*Disconnect-Connection-To-Network*	✓				✓	
Denial of Service	*DeleteObject*	✓	✓			✓	
Flooding	*Who-is-Router-to-Network*	✓				✓	
Flooding	*Reinitialize-Device*	✓				✓	
Flooding	*I-am*	✓				✓	
Flooding	*Any Malformed Packet*	✓				✓	
Malformed Broadcast	*Reject-Message-To-Network*	✓				✓	
Malformed Broadcast	*CreateObject*	✓				✓	
NetworkLoop	*InitializeRoutingTable*	✓				✓	
Reconnaissance	*NPDU probes*				✓		
Reconnaissance	*Read Property*				✓		
Reconnaissance	*Whois*				✓		
Reconnaissance	*Whoami*				✓		
Reconnaissance	*Who-is-Router-to-Network*				✓		
Reconnaissance	*WhoHas*				✓		
Routing table attack	*InitializeRoutingTable*					✓	
Shutdown/Reboot	*Reinitialize-Device*					✓	
Smurf attack	*source address manipulation*	✓				✓	
Spoof device	*I-am*	✓	✓		✓		
Traffic Redirection	*I-am-Router-to-Network*	✓			✓		
Traffic Redirection	*Router-Available-to-Network*	✓			✓		
Write Attack	*Write Property*		✓			✓	

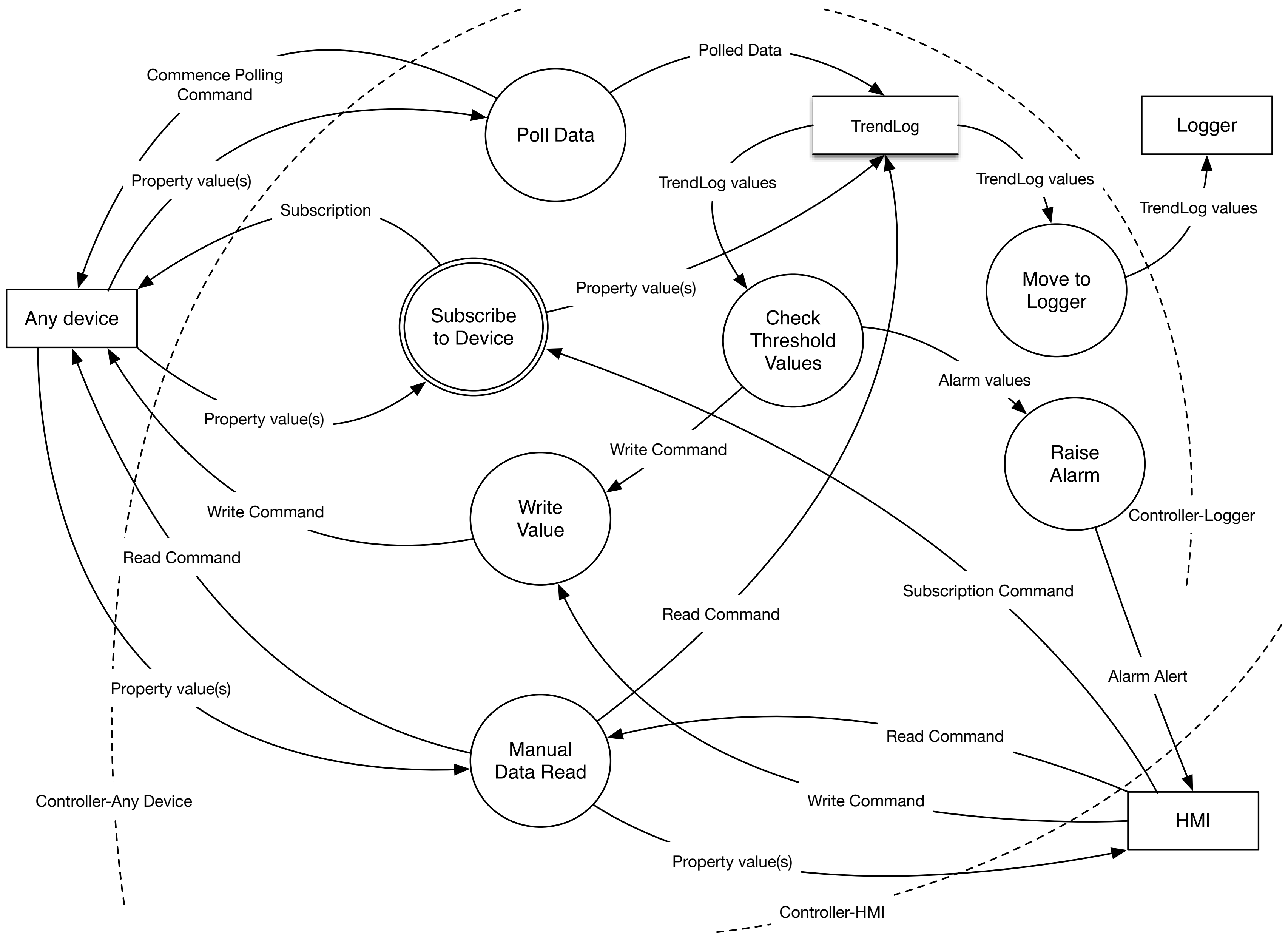


Figure 4.3: DFD of controller actions in a BACnet managed HVAC scenario, replicated from Peacock, Johnstone and Valli (2018, p267)

The appropriate STRIDE threat classes for each unique element in the model was defined, and correlated with the STRIDE threat classification of known vulnerabilities presented in Table 4.6 (Peacock et al., 2018). The threats which each identified attack could initiate towards each element was mapped, a subset is detailed in Table 4.7. To rank the impact of each attack against the scenario, a count of threats against elements was used. The total threats posed by all noted attacks was 652. Of the 652 total threat instances, Denial of Service was the largest threat class with 306 instances.

Table 4.7: Identified BACnet specific attacks, classified by the STRIDE matrix, replicated from Peacock, Johnstone and Valli (2018, p269)

Element	A1	A2	A6	A7	A11	A13	A14	A20	A21	A22	A23	A24	A26
Alarm Alert	T,D	T,D	T,D	D	D	D	I	D	D	D	T,I	I	T,D
Alarm Values	T,D	T,D	T,D	D	D	D	I	D	D	D	T,I	I	T,D
Any Device	S	S	S	S	S	S	-	-	-	S	S	S	-
Check Threshold Values	T,D	T,D	T,D	D	D	D	I	D	D	S,D	T,I	I	T,D
Commence Polling Command	T,D	T,D	T,D	D	D	D	I	D	D	D	T,I	I	T,D
HMI	S	S	S	S	S	S	-	-	-	S	S	S	-
Logger	S	S	S	S	S	S	-	-	-	S	S	S	-
Manual Data Read	T,D	T,D	T,D	D	D	D	I	D	D	S,D	T,I	I	T,D
Move to Logger	T,D	T,D	T,D	D	D	D	I	D	D	S,D	T,I	I	T,D
Poll Data	T,D	T,D	T,D	D	D	D	I	D	D	S,D	T,I	I	T,D
Polled Data	T,D	T,D	T,D	D	D	D	I	D	D	D	T,I	I	T,D
Property Value(s)	T,D	T,D	T,D	D	D	D	I	D	D	D	T,I	I	T,D
Raise Alarm	T,D	T,D	T,D	D	D	D	I	D	D	S,D	T,I	I	T,D
Read Command	T,D	T,D	T,D	D	D	D	I	D	D	D	T,I	I	T,D
Subscribe to Device	T,D	T,D	T,D	D	D	D	I	D	D	S,D	T,I	I	T,D
Subscription	T,D	T,D	T,D	D	D	D	I	D	D	D	T,I	I	T,D
Subscription Command	T,D	T,D	T,D	D	D	D	I	D	D	D	T,I	I	T,D
TrendLog	T,D	T,D	T,D	D	D	D	I	D	D	D	T,I	I	T,D
TrendLog Values	T,D	T,D	T,D	D	D	D	I	D	D	D	T,I	I	T,D
Write Command	T,D	T,D	T,D	D	D	D	I	D	D	D	T,I	I	T,D
Write Value	T,D	T,D	T,D	D	D	D	I	D	D	S,D	T,I	I	T,D

For each individual attack, the total impact of each threat class against the scenario was derived, Table 4.8 outlines the full threat counts for each attack. Using this classification, the commands which have the highest threat potential for legimiate malicious action can be obtained. Six attacks were identified as equal highest with 39 threat counts, these attacks used application layer commands *Subscribe-CoV*, *I-am* and *DeleteObject*, and network layer commands relating to BACnet routers. Identifying legitimate malicious instances of these commands is dependent on the context derived from the individual implementation of the BACnet system. Exploration of identifying these attacks is undertaken in §4.6.9.

Object and service analysis against a number of retrieved devices is undertaken in §4.6.7. The details of these devices were correlated with the classified attacks described in Table 4.6. Of the

Table 4.8: Total threat counts based on known attacks against scenario model, replicated from Peacock, Johnstone and Valli (2018, p270)

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26	Total
S	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0	0	0	0	0	0	0	10	3	3	3	0	58
T	18	18	18	18	0	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0	18	126
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	0	0	0	0	18	18	18	18	18	18	0	0	0	18	18	18	0	162
D	18	18	18	18	18	18	18	18	18	18	18	18	18	0	0	0	0	0	0	18	18	18	0	0	0	18	306
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Total	39	39	39	39	21	39	21	21	21	21	21	21	21	18	18	18	18	18	18	18	18	28	39	21	21	36	652

known attacks against BACnet, the occurrence of surveyed devices which implement the affected services are identified, see Table 4.9

It follows that the most common threats to devices are those which use the most common services to undertake malicious action. Thus, the most common services, *Read property*, *Who-is*, *I-am*, *Who-has*, *I-have* and *Write property* which are contained in over 97.8% of the devices analysed are of interest. As these are the most common services in use, it is difficult to undertake binary classification between malicious and normal commands. Therefore, further analysis of these service types in the network traffic is required to classify if a malicious-command is being undertaken, as opposed to normal operations on the BAS.

Sensors and actuators are clearly the most limited devices in terms of attempting to attack other devices.The majority of sensor and actuator devices do not operate the client side of each service, which allows a device to initiate functions in other devices. Sensor and actuator devices, as designed, can execute write and read commands, allowing for other non-sensor devices to instruct sensor devices to maliciously write values into its data structures.

Controllers, advanced controllers and workstations, by design have both the server and client side services available, which provides these devices with the means to control other devices. Thus, an attacker has more control over a whole system if he/she has control of a workstation or controller-type device.

Table 4.9: Comparison of extracted device services, and occurrence in devices to classified known threats

BIBBS	Attacks	Sensor Percent	Controller Percent	Actuator Percent	advController Percent	Workstation Percent
AE-AVM-A	Denial of Service, Malformed Broadcast, Reconnaissance, Write Attack	0.00	0.00	0.00	0.00	4.17
AE-N-E	Reconnaissance	0.00	0.00	0.00	7.41	8.33
AE-VM-A	Reconnaissance, Write Attack	0.00	0.00	0.00	0.00	100.00
DM-ADM-A	Reconnaissance	0.00	0.00	0.00	0.00	79.17
DM-ANM-A	Reconnaissance, Flooding, Spoofing	0.00	0.00	0.00	0.00	70.83
DM-BR-A	Denial of Service, Malformed Broadcast	0.00	0.00	0.00	0.00	25.00
DM-BR-B	Denial of Service	0.00	6.64	0.00	40.74	16.67
DM-DDB-A	Reconnaissance, Flooding, Spoofing	7.14	24.22	0.00	96.30	100.00
DM-DDB-B	Flooding, Reconnaissance, Spoofing	85.71	100.00	100.00	100.00	100.00
DM-DOB-A	Reconnaissance	0.00	1.56	0.00	25.93	37.50
DM-DOB-B	Reconnaissance	85.71	100.00	94.12	98.15	100.00
DM-OCD-A	Denial of Service, Malformed Broadcast	0.00	0.00	0.00	0.00	16.67
DM-OCD-B	Denial of Service, Malformed Broadcast	0.00	2.34	0.00	18.52	20.83
DM-RD-A	Denial of Service	0.00	0.00	0.00	0.00	37.50
DM-RD-B	Denial of Service	35.71	53.52	29.41	96.30	45.83
DS-AM-A	Write Attack	0.00	0.00	0.00	0.00	12.50
DS-AV-A	Reconnaissance	0.00	0.00	0.00	0.00	16.67
DS-COV-A	Denial of Service	0.00	7.03	0.00	27.78	70.83

continuation of Table 4.9

BIBBS	Attacks	Sensor Percent	Controller Percent	Actuator Percent	advController Percent	Workstation Percent
DS-COV-B	Denial of Service	21.43	42.97	64.71	79.63	16.67
DS-M-A	Write Attack	0.00	0.00	0.00	0.00	100.00
DS-RP-A	Reconnaissance	0.00	14.06	0.00	51.85	100.00
DS-RP-B	Reconnaissance	100.00	100.00	100.00	100.00	100.00
DS-V-A	Reconnaissance	0.00	0.00	0.00	0.00	100.00
DS-WP-A	Write Attack	0.00	13.67	0.00	55.56	100.00
DS-WP-B	Write Attack	100.00	100.00	100.00	100.00	66.67
NM-RC-B	Denial of Service, Flooding, Traffic Redirect	0.00	0.78	0.00	0.00	0.00
SCHED-AVM-A	Denial of Service, Malformed Broadcast, Reconnaissance, Write Attack	0.00	0.00	0.00	0.00	4.17
SCHED-E-B	Reconnaissance,Write Attack	0.00	0.00	0.00	29.63	16.67
SCHED-I-B	Reconnaissance, Write Attack	0.00	3.91	0.00	92.59	16.67
SCHED-R-B	Reconnaissance	0.00	1.56	0.00	0.00	0.00
SCHED-VM-A	Reconnaissance, Write Attack	0.00	0.00	0.00	0.00	100.00
SCHED-WS-A	Reconnaissance, Write Attack	0.00	0.00	0.00	0.00	33.33
SCHED-WS-I	Reconnaissance, Write Attack	0.00	2.34	5.88	0.00	0.00
T-AVM-A	Denial of Service, Malformed Broadcast, Reconnaissance, Write Attack	0.00	0.00	0.00	0.00	12.50
T-VMT-E	Reconnaissance	0.00	0.00	0.00	5.56	16.67

4.3.1 Attack modelling

As identified in section 4.6.5, there exists potential Denial of Service methods in BACnet's priority array implementation. Modelling the priority array was undertaken to identify approaches to detect contextually abnormal commands, and improve understanding of the security requirement of the bounded array. The *Anylogic* simulation software (AnyLogic, 2017) was used to model the priority level security vulnerability, described in Section 4.1.2. A conceptual model of the problem space was developed, detailed in Figure 4.5). From this model, a dynamic agent-based simulation model, capable of representing the identified problem with objects and their associated properties in a BACnetwork was generated.

A typical run of the simulation is outlined in Figure 4.4. An object can be connected to the priority list at any priority level, as per the BACnet specification (SSPC-135, 2012). Results from two devices writing to a property at the same priority level in the simulation model is detailed in Table 4.10.

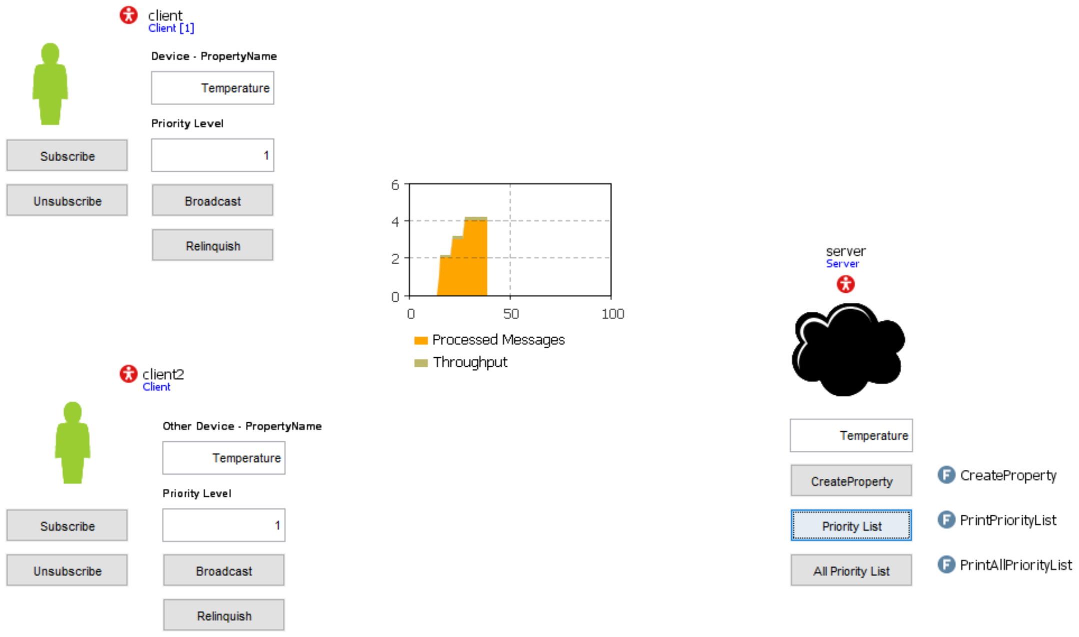


Figure 4.4: Sample simulation run of BACnet bounded priority array, replicated from Peacock, Johnstone and Valli (2018, p264)

Modelling the priority array has identified that the Denial of Service issue can exist, namely, two devices can write to the same priority level in the array, with the second device overwriting the first device's value (Peacock et al., 2018). Due to the variance of BACnet implementations, described in §4.1, this behaviour may not be representative or definitive for every BACnet device. In a generic device, defined by the standard, this issue exists when represented as a theoretical model.

Table 4.10: Simulated array before and after two devices write to a property at priority 1, adapted from Peacock, Johnstone and Valli (2018, p265)

Array after First Device Write			Array After Other Device Write		
Priority level	Value	Identifier	Priority level	Value	Identifier
1	11	Device	1	68	Other Device
2	NULL		2	NULL	
3	NULL		3	NULL	
4	NULL		4	NULL	
5	NULL		5	NULL	
6	NULL		6	NULL	
...			...		
16	NULL		16	NULL	

4.3.2 Priority array attack implementation

The *BACnet open stack* (Karg, 2017) was used to test the priority array issue. The procedure used to test involved the following process.

1. Device A reads the base value of the Present value property of Device B
2. Device A writes an initial value to Priority array 12 on Device B
3. Device A reads the Present value property of Device B
4. Device A reads the Priority Array of Device B
5. Device C writes a different value to Priority Array 12 on Device B
6. Device A reads the Present value property of Device B
7. Device A reads the Priority Array of Device B
8. Device C writes a relinquish command to Priority array 12 on Device B
9. Device A reads the Present value property of Device B
10. Device A reads the Priority Array of Device B

Table 4.11 presents the values from this procedure. As can be seen, upon relinquishing the value written into priority 12 on Device B, the original value written by Device A is lost. Therefore, when two devices write to the same priority level in a third device, the last write wins scenario occurs. Although the priority array allows for up to 16 devices to write values to the same property, each of these devices must write to a different priority array value, else a value is lost. Given there are no restrictions on which devices may write at specific priorities to devices, the priority array can be overwritten.

Table 4.11: Terminal output of priority array value overwrite testing

Source Device	Command	Output
Device A	./bin/bacrp 120 1 1 85	0
Device A	./bin/bacwp 120 1 1 85 12 -1 4 20.0	WriteProperty Acknowledged!
Device A	./bin/bacrp 120 1 1 85	20
Device A	./bin/bacrp 120 1 1 87	{NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, 20.000000,NULL, NULL, NULL, NULL}
Device C	./bacwp 120 1 1 85 12 -1 4 50.0	WriteProperty Acknowledged!
Device A	./bin/bacrp 120 1 1 85	50
Device A	./bin/bacrp 120 1 1 87	{NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, 50.000000,NULL, NULL, NULL, NULL}
Device C	./bacwp 120 1 1 85 12 -1 0 0	WriteProperty Acknowledged!
Device A	./bin/bacrp 120 1 1 85	0
Device A	./bin/bacrp 120 1 1 87	{NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL}

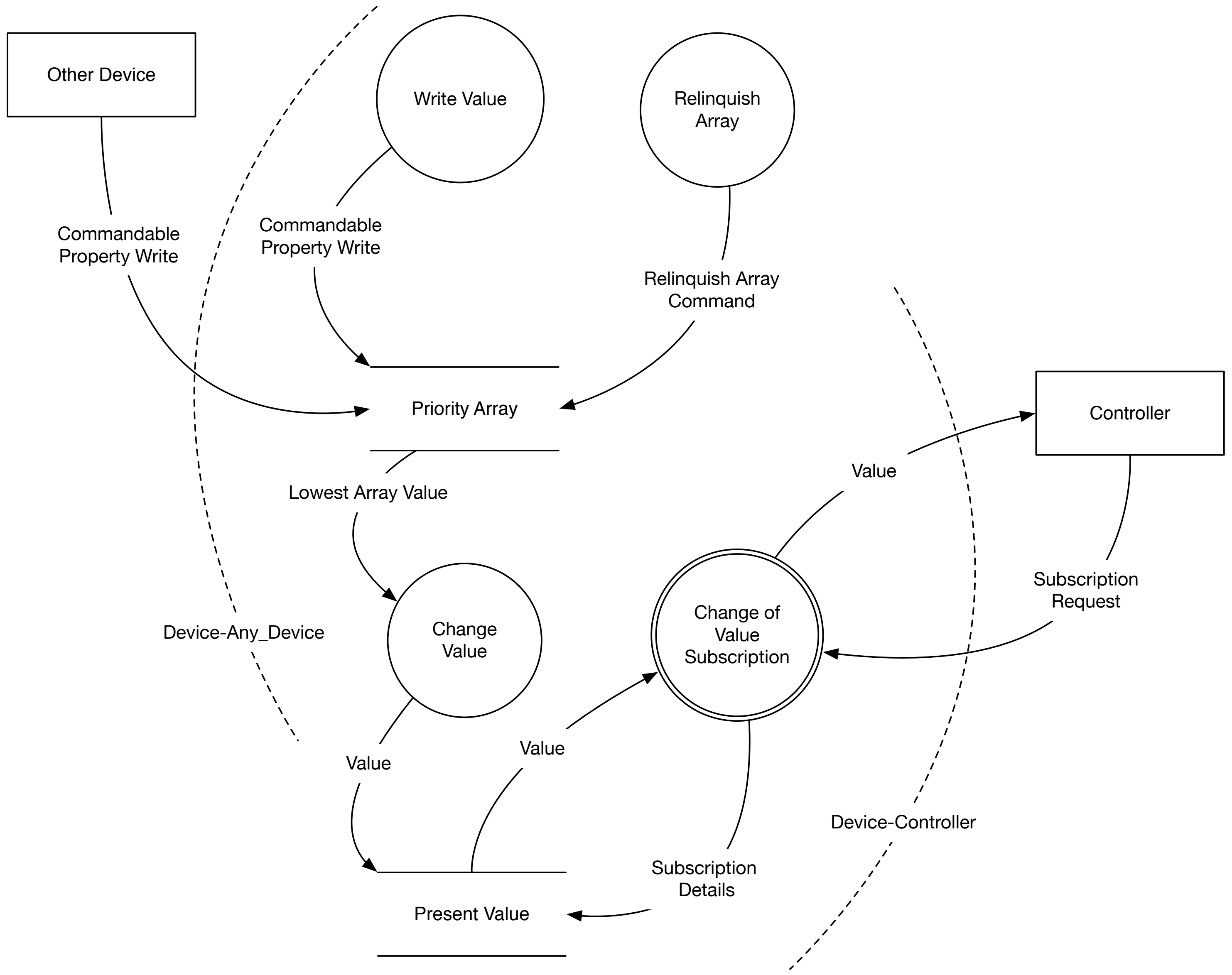


Figure 4.5: DFD of bounded array priority process, replicated from Peacock, Johnstone and Valli (2018, p264)

4.4 Assessing existing simulation environments

There is a lack of available network data for security analysis in BAS, particularly for BACnet. However, the means for generating network traffic exists in the form of the *BACnet open stack* (Karg, 2015), and a number of commercial simulation environments (Scada Engine, 2009a; CBMS, 2015b). To undertake analysis of intrusion detection techniques with machine learning, a large amount of data is required dependent on the algorithm selected. To trial the generation of data, and prototype the identified attacks, two smaller, focused datasets were implemented, classified as Trial dataset and Theory testing dataset. The use of smaller datasets allowed for initial testing of algorithms to be undertaken. Additionally, the prototyping of dataset generation allowed for short-comings in existing BACnet simulation software to be identified, and determine the requirements for a larger simulation, which is discussed in §4.6.3. Datasets were generated based on specific scenarios, which incorporated a base BAS, and an implemented attack.

4.4.1 Trial Dataset: Write Attack

The Trial Dataset was designed as a minimal partial HVAC system, which consisted of one temperature sensor, one fan and one controller device which communicated actions to the fan, based on the sensor's provided data readings. In accordance with HVAC design practices, fan speed changes occurred at a minimum 15 second interval, to replicate time delays present to prevent fan drive damage (Stanford III, 2011). The attack defined for the Trial Dataset was a dormant threat in the controller which would pseudo-randomly interact with the fan. The attacker sent legitimate fan speed change commands in quick succession of each other, with the intent to cause physical damage to the fan. The generated Trial Dataset contained three hours of BACnet simulated data in network capture format, equating to 25,000 frames. The topology of the simulation environment is shown in Figure 4.6. The BACnet simulation tool *SCADAengine* (Scada Engine, 2009a) was used for the simulated temperature sensor and fan, The *BACnet open stack* (Karg, 2015) acted as the controller, receiving temperature readings from the simulated temperature sensor, and sending fan speed commands to the simulated fan.

4.4.2 Trial Dataset: ANN approach

As an initial test of supervised machine learning, a backpropagation Artificial Neural Network (ANN) was selected to identify the write attack. The ANN implementation is detailed as Equation 4.1 - Equation 4.4.

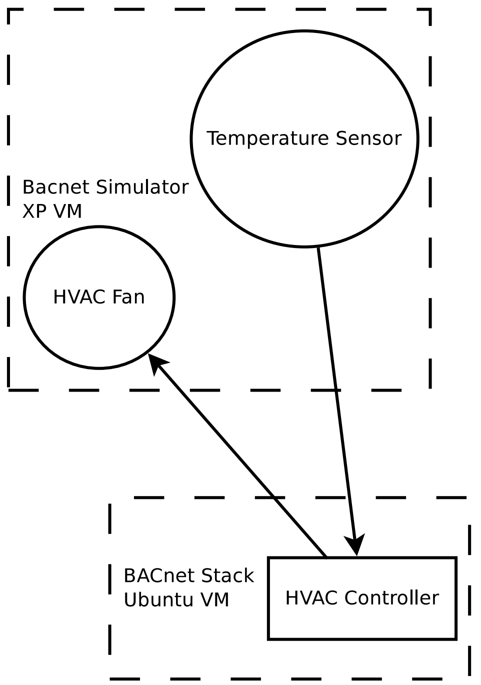


Figure 4.6: Network topology for Trial Dataset experiments, replicated from Johnstone, Peacock and den Hartog (2015, p60)

$$s_j = \sum_{i=1}^{n} w_i x_i$$

where (Equation 4.1)

$x_i =$ Input i

$w_i =$ Weighting applied to x_i

$$o_j = \phi(s_j + b_{ij})$$

where

$o_j =$ Output j (Equation 4.2)

$b_{ij} =$ Bias factor applied to node s_j of (Equation 4.1)

$\phi =$ The activation function

$$\delta_j = e_j \phi'_j(s_j) \qquad \text{(Equation 4.3)}$$

$$\delta_j = -\frac{\partial \mathcal{E}}{\partial y_j}\phi'_j(s_j)$$

where (Equation 4.4)

Neuron j is hidden

e = error term

Pre-processing of the dataset involved removing all but the *write command* frames, which were paired into events. The pairing resulted in 1,000 write events which formed the final dataset. The Delta time (Δt) between same frames (writes) are the events of interest, as each frame contains a legitimate command with legitimate values.

The first 500 events were used to train the ANN. The remaining 500 events were used to test the effectiveness of the trained ANN in classifying previously unseen events. The ANN was optimised through testing the number of iterations, the number of hidden layers and the learning rate. The number of hidden layers were enumerated from one through eight. Testing began with a simple three-layer network, which held one hidden layer. The results were encouraging, with a training time of 6.349 seconds and a classification accuracy of 90.4% against the test data, with a classification time of 0.005 seconds. The ANN was then extended to four layers. In the two hidden layer ANN, the training time was 5.718 seconds with a classification accuracy of 100.0% (classification time 0.006 sec.). With three hidden layers, the training time was 2.059 seconds and had a classification accuracy of 100.0% (classification time 0.007 sec.). Figure 4.7 shows that both three and five hidden layers appear to be optimal, the classification time is longer for five hidden layers (0.008 sec.).

Given that Backpropagation is a gradient descent method, it is possible the ANN to not perform well if it does not converge on the global error minimum. The error rate was examined to ensure that the ANN was not being trapped in a non-optimal local minimum, detailed in Figure 4.8. Table 4.12 shows that the training time for the ANN increases linearly with the number of iterations, this is not an issue provided training time is not the rate determining step. There was a 10% decrease in classification accuracy when the learning rate was varied from 1.0 to 0.1. The learning rate is a measure of the size of the step taken down the gradient, so large values of learning rate correspond to smaller steps (which would be more accurate, but take longer to converge).

4.4.3 Trial Dataset: Flow Analysis

An alternative approach to using the ANN was undertaking flow analysis. As the write attack described in section §4.4.2 is a frequency attack, identifying increases in frequency can be used for detection. The network commands sent over the network were converted into a time series line plot with a bin size of one minute, detailed in Figure 4.9. Through frequency comparisons for each command, it can be seen that *Who-is* and *I-am* command frequencies peak when *Write property* and *Read property* commands peak in phase. These relationships exist due to the implementation

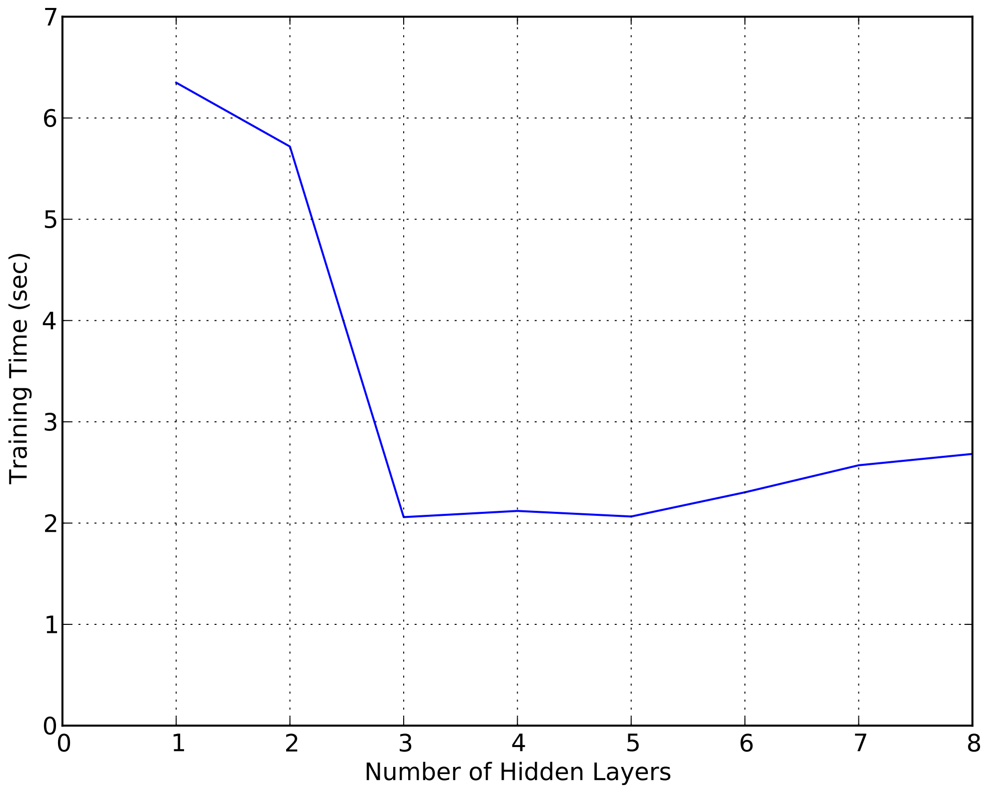


Figure 4.7: Number of hidden layers compared to the training time, replicated from Johnstone, Peacock and den Hartog (2015, p61)

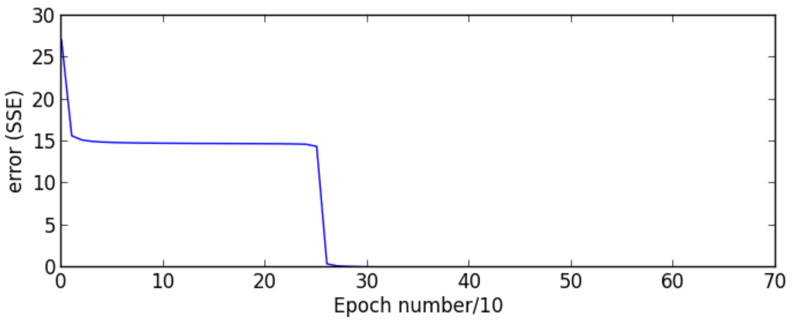


Figure 4.8: Error convergence in the ANN, replicated from Johnstone, Peacock and den Hartog (2015, p62)

Table 4.12: Iterations of the ANN vs. Training Time of the Network, replicated from Johnstone, Peacock and den Hartog (2015, p62)

Iterations	Time (sec.)
100	0.875
200	1.710
300	2.593
400	3.522
500	4.302

of the network devices which do not hold network addresses, thus when a *Write property* or *Read property* command occurs, the address of the device must be requested (*Who-is*) and received (*I-am*). An improved dataset which holds network addresses would reduce this phenomenon.

Given we are looking for bursts of traffic over a short duration, further reducing the bin sizes reveals the anomalous command events on the network. A comparison of bin sizes for write commands are presented in Figure 4.10. Further analysis could identify a specific host generating this traffic. While this analysis is useful for this specific attack, it requires contextual knowledge of the normal network interactions for the attack to be identified. Further, non-frequency attacks may not have an identifiable change in phase. Regardless, this method may be complementary to other, more process intensive methods such as machine learning, and thus was explored further.

4.4.4 Theory Testing Dataset: CoV attack

As described in §4.3.1, a dataset was generated to undertake testing of the theoretical CoV attack. Similar to the Trial Dataset, the Theory Testing Dataset represents a subset of a BACnet controlled HVAC system, with the additional CoV reporting functionality. The Theory Testing Dataset contains a thermostat, and an air handling unit (AHU) controller device, see Figure 4.11. The thermostat Raspberry Pi used a *CBMS* BACnet server instance (CBMS, 2015b), while the interaction from the controller device was implemented using the *BACnet open stack* (Karg, 2015). The configuration of the thermostat required an additional *CBMS* engineering tool (CBMS, 2015a), implemented on a Windows 7 machine. A controller acts as a client, subscribing to the thermostat *present_value* property. When a value change occurs over a threshold value, the controller is notified (Peacock et al., 2018).

The scenario for testing the attack consisted of a a malicious client device subscribing to the thermostat server device. When subscribing, the malicious client can set the timeout value and retry attempts in the case of a device losing connection. For this test, the wait time is set to 10 seconds, and the retries are set to 3. Next, the malicious client disconnects from the network. While disconnected, the subscribed value changes multiple times on the thermostat device. After a period of time, the client reconnects to examine the behaviour. While the client is disconnected, the server device attempts to send the *CoV* notifications, and waits the defined timeout and retry times. The aim of creating this dataset was to determine if the client device received delayed notifications of the server value changes after reconnecting to the network.

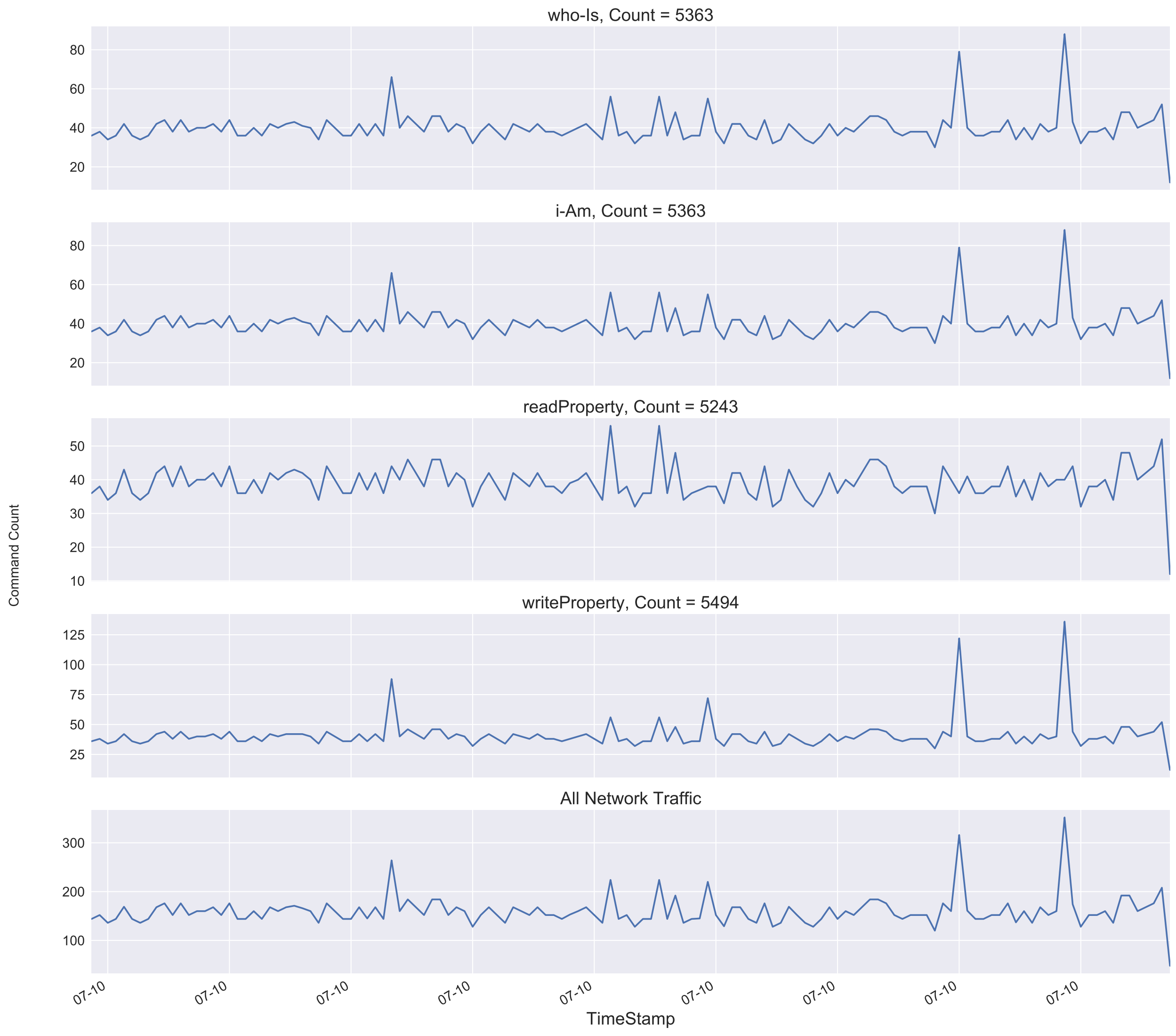


Figure 4.9: Frequency breakdown of commands in the Test Dataset, with bin size 1 minute

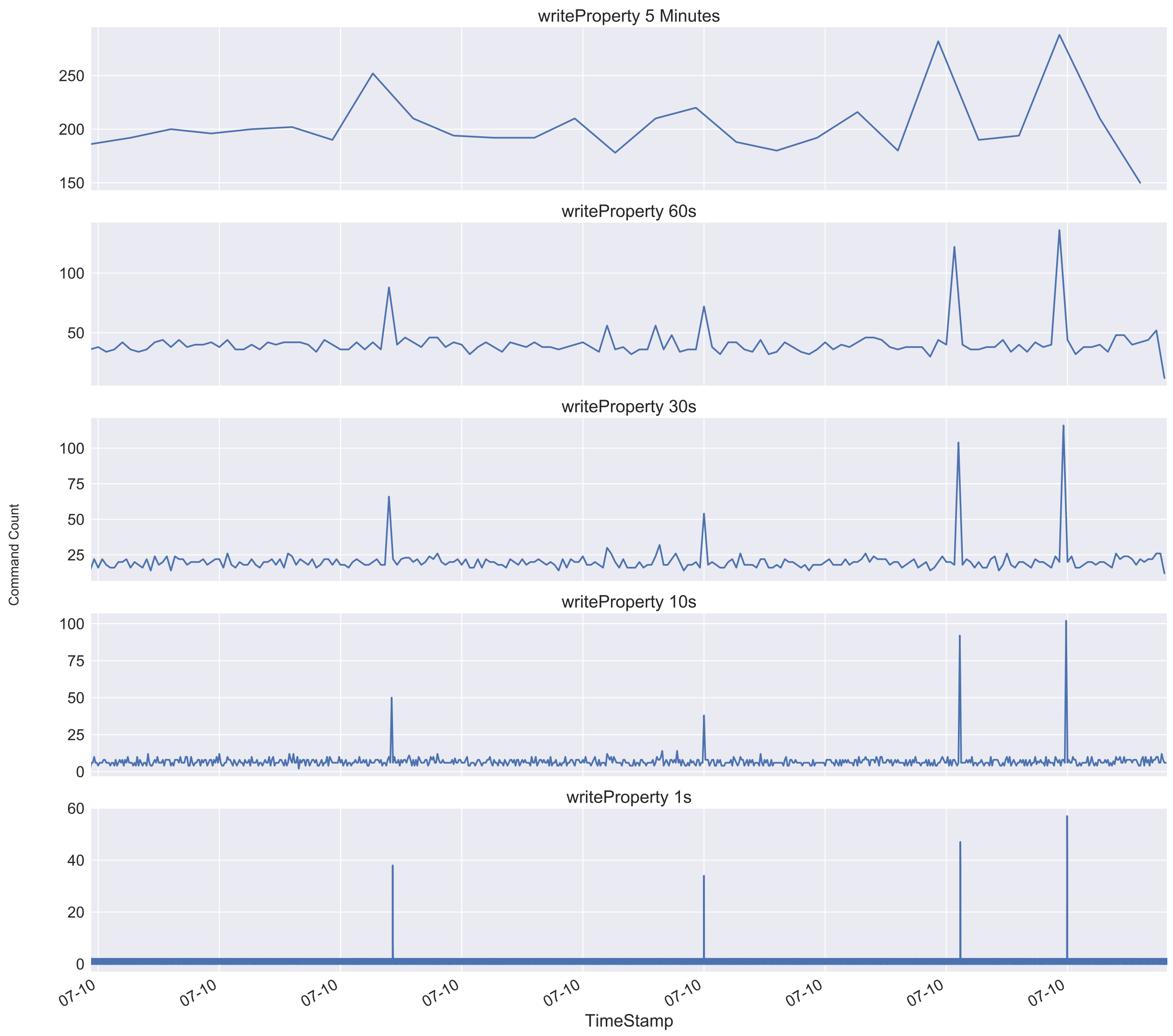


Figure 4.10: Comparison of bin sizes for identifying *Write property* command frequencies

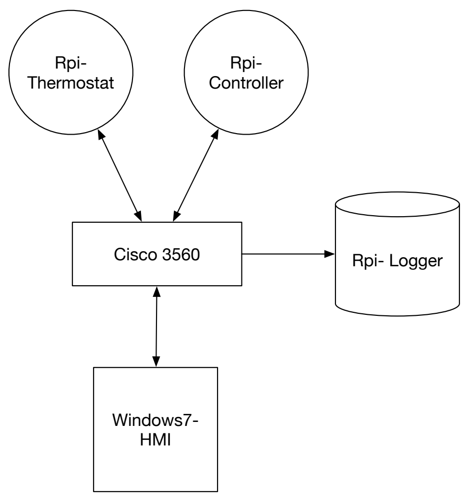


Figure 4.11: Experimental simulation setup for Theory Testing Dataset, replicated from Peacock, Johnstone and Valli (2018, p261)

4.4.5 Theory Testing Dataset: Analysis

Table 4.13 outlines an excerpt of the captured behaviour, outlined in (Peacock et al., 2018). Due to a limited network handler implementation, the normal response from a confirmed request is a reject response. Naturally, when the malicious device disconnects, the server can no longer communicate. While attempting to communicate, the server waits 10 seconds for a response, and then resends the confirmation twice, for three total confirmations. During the 30 second disconnected window, the subscribed server value was changed multiple times, none of these changes triggered a further notification while the server waited for the previous changes acknowledgement packet. Entry 81438 and 81439 in Table 4.13 detail a normal notification and ACK, after a value change and the malicious client device was reconnected. When the malicious client reconnected, the changes that occurred while waiting for the timeouts and retries were never disseminated to the client (Peacock et al., 2018).

From the initial experiment and testbed implementation, the extent of the attack is unclear. Further experimentation was required to implement additional devices, to determine if the attack prevents the server device from communicating to other subscribed devices.

4.5 Simulator assessment and initial attack implementation outcomes

For the simple write attack, the context of the command is of importance. The contextual information in this case, is the Δt between write commands occurring. By explicitly distinguishing this feature, the ANN model was able to detect the context of the network command, and thus identify the attack. More complex network scenarios would require additional learning for these contextual details, which can be revealed through the use of temporal modelling, such as time series analysis

Table 4.13: Initial experimentation results, replicated from Peacock, Johnstone and Valli (2018, p262)

No.	Time	Source	Destination	Protocol	Length	Info
80040	19:58:32.5	192.168.1.12	192.168.1.5	BACnet	85	Confirmed-REQ confirmedCOVNotification[67] device, 9999 analog-output, 1 present-value
80041	19:58:32.5	192.168.1.5	192.168.1.12	BACnet	60	Reject unrecognized-service[67]
80327	19:58:48.4	192.168.1.12	192.168.1.5	BACnet	85	Confirmed-REQ confirmedCOVNotification[69] device, 9999 analog-output, 1 present-value
80510	19:58:58.4	192.168.1.12	192.168.1.5	BACnet	85	Confirmed-REQ confirmedCOVNotification[69] device, 9999 analog-output, 1 present-value
80656	19:59:08.4	192.168.1.12	192.168.1.5	BACnet	85	Confirmed-REQ confirmedCOVNotification[69] device, 9999 analog-output, 1 present-value
81438	19:59:48.5	192.168.1.12	192.168.1.5	BACnet	85	Confirmed-REQ confirmedCOVNotification[70] device, 9999 analog-output, 1 present-value
81439	19:59:48.5	192.168.1.5	192.168.1.12	BACnet	60	Reject unrecognized-service[70]

or other machine learning algorithms.

As demonstrated, the basic time series analysis against the trial dataset was useful for identifying the contextual write attack. Given the attack was frequency based, methods such as graph analysis, and packet counting could be useful for either detection, or building the context around normal network commands traversing the network between hosts. There may be limitations however, particularly if the distinguishing features between legitimate and malicious commands have a closer value range. The ANN applied to test the dataset provided justification to pursue applying additional machine learning algorithms to the BACnet traffic classification problems. The CoV proof of concept (POC) implementation in the theory testing dataset described the issues identified in the protocol specification review. A larger, virtual testbed was deemed necessary which could incorporate further devices to test the effect of the attack.

The current version of the *SCADAengine* BACnet simulator (Scada Engine, 2009a) had many limitations, primarily, lack of value generation or non-linear value stepping, and the requirement of Windows XP. The *CBMS* server device (CBMS, 2015b) used for the theory testing dataset also had a restrictive value generation and value stepping implementation. Additionally, *SCADAengine* (Scada Engine, 2009a) and *CBMS* (CBMS, 2015b) are closed source stacks, in which it is difficult to implement missing features. Both *SCADAengine* (Scada Engine, 2009a) and *CBMS* (CBMS, 2015b) provided the ability to store BACnet data structures and undertake limited network interactions. However, for a larger, more robust simulation, further programmatic network interaction was required. As such, given both *SCADAengine* (Scada Engine, 2009a) and *CBMS* (CBMS, 2015b) are closed source, and hardware restrictive, they were not used further in the research.

Comparatively, the flexibility of the *BACnet open stack* (Karg, 2015) was promising for generating a larger testbed. Sample device templates existed in the stack which could be adapted to the scenario, to be used for data storage and initiate network communications. Due to its ability to have missing features implemented, and the ability to pass generated values to BACnet data structures, the *BACnet open stack* was selected for further simulation of BACnetworks for the research.

Lessons learned from the implementations of both simulators identified the requirements for developing a larger testbed. Specifically, a means of generating values to send over the network, the ability to construct BACnet devices, and storage of network addresses for devices. Further, investigating the network footprint of a real BACnet/IP network was deemed useful to ground the simulation scenario.

4.6 Data collection and generation

Section 4.6 describes the rationale of both collecting real and generating synthetic network traffic for the purpose of anomaly detection. First, the real network data collection is described, followed by initial analysis of the captured network data. Next, a description of the defined building simulation scenario, physical data generation and network design for data generation is presented. Finally, the designed attack framework is defined, with testing of each defined attack described.

4.6.1 Real data collection

Real BAS data was collected from an Australasian University for the study, with the intention of being used for validation of the simulation implementation. Data capturing involved port mirroring of the BAS VLAN for one building on campus. The building consisted of a number of network visible controllers operating over BACnet/IP, with underlying sensors, actuators and application specific controllers operating over serial connection using BACnet/MS-TP. One month of network stream was collected, with network activity averaging 1GB per day, consisting of over one million frames. The network capture was on the backbone/management level of the topology, as such, it was not possible to capture sensor data for the building. However, the insights gained from this dataset, discussed in §5.2.1 were used to define some interactions in the simulation.

4.6.2 Analysis and validity

Initial analysis of the network data was undertaken using *Moloch* (AOL, 2017), a network stream visualisation and analysis tool. From this software per-host packet counts were obtained in addition to visual flow structures of the traffic. It was revealed that the traffic is extremely regular, as the majority of traffic is automated, based on time. Upon further analysis, it was determined that over

78% of the traffic captured was not passing over the default BACnet port. Rather, the vendor of the controller uses a proprietary middleware protocol to communicate with BACnet devices for point configuration over a separate communication port. Point configuration, in this context consists of *Write property* commands to devices. This poses a number of questions for intrusion detection in BACnet-managed BAS. Given a portion of attacks operate using write commands, when a write command is passed over the default BACnet port when running controllers from specific vendors, it would be classed as anomalous, as write commands operate on the proprietary port. This reinforces the requirement of learning each specific BAS network's traffic profile for intrusion detection, rather than developing a catch-all intrusion detection system for BAS. Thus, even though BACnet is an open source protocol, there are fundamentally different proprietary elements dependent on the vendor type of controllers and implementation. In addition to the wide range of technologies, device versions and control strategies, a generic detection method does not seem suitable for all BAS implementations. Further research is required to investigate vendor middleware protocols to determine how they interface with BACnet, given the lack of network traffic dissectors, as attacks could be propagated over the network using these middleware interfaces, if the protocol is reverse engineered.

The original intention of collecting the real network data was to use this data for statistical comparison to the synthetically-generated data for simulation validity and justification. Every implementation of a BACnetwork is tailored to the specific building design. As such, each implementation is different, given the wide range of devices, vendors and physical media which can incorporate BACnet. Given that the research specifically looks at BACnet/IP, there are some limitations to the real dataset collected. First, the design of the real buildings network does not have all devices using BACnet/IP, rather only the controllers. All field devices used serial communications from which it was not capable of capturing network traffic. Second, given the choice of vendor, some of the network traffic was not dissectible due to a proprietary middleware implementation of the BACnet protocol operating on the network. Therefore, a comparison between a network, which only has one type of device, and utilises vendor specific protocol implementations compared to a full BACnet/IP network was not a logical course of action. As such, the simulation design used for the research was defined using a range of sources, discussed in §4.6.3.

4.6.3 Building simulations

There are two common use cases for simulation in building automation systems. One is network analysis and protocol modelling, the focus of this research and other cyber security based projects mentioned in §2. The second is for the building design domain, aimed at improving energy efficiency through whole building modelling and simulation. A range of simulation and modelling software exist for both use cases. For BACnet, network simulation software include *SCADAengine* (Scada Engine, 2009a) and the *BACnet open stack* (Karg, 2017). Typically, data generated in these simulators are simplistic, with the aim to generate network traffic rather than have meaningful data inside these network transactions.

For building design, simulations are used to generate data for load calculations, to give an understanding of energy impact and building material longevity. Thus, simulation software and languages, such as *Modelica* (Modelica, 2018) and *Energyplus* (EnergyPlus, 2018) used by building designers, are focused on accurate data and energy calculations, with network traffic rarely, if ever, implemented. For this research, the primary interest is network simulation, however, the data held in network transactions are of interest for the context of feature selection. While the data would ideally be realistic, the estimations generated by building design simulations have a high degree of variability compared to real-world conditions. The variability can be accounted for by the modeller's bias, or underestimating the impact of occupant and heat emitting equipment operating within the building spaces. As such, the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) define a building design model for an existing building to be calibrated if it falls within $\pm10\%$ of mean bias error and the coefficient of variation root mean squared error falls within $\pm30\%$ using hourly data, or $\pm5\%$ using monthly data (ASHRAE, 2012b).

For modelling the internal systems of a building for energy simulation, a range of variables are used, including local weather readings, occupancy models, building geometry details, construction materials, internal equipment load, building occupancy schedule and HVAC type. Similarly, for simulating a BACnetwork, appropriate digital representation of devices is required, in addition to protocol defined network transactions.

Therefore, the simulation environment required for this research takes building design variables into consideration, when developing a simulation scenario, to generate a simulation which accounts for real-world data loads, as opposed to generating non-meaningful data for traffic analysis.

4.6.4 Simulation design

There are a number of interrelated parts operating as a control loop structure within BAS. Four core requirements were identified for the generation of the simulation:

1. A system scenario;
2. Generation of environmental variables;
3. Definitions of devices;
4. Network communications.

These four requirements are split into corresponding simulation levels. The first level is the definition of a system scenario, which defines the user, and interaction of components in the other three levels. The system scenario is defined in Section 4.6.5 The second level, defined as the physical level describes how synthetic values will be representative of environmental phenomena. For this to occur, a range of algorithms used in building energy simulation systems and HVAC engineering were implemented, along with a range of core assumptions. Section 4.6.6 describes in detail the physical level. The third level is the device level, which describes the BACnet representation of devices which hold the synthetic data, and propogate the data around the system. Section 4.6.7

describes this level. The fourth and final level is the network level, which outlines the network topology for the simulation, and simulates data traversal over the network dependent on the communication processes implemented between devices, described in Section 4.6.8. Additionally, the network level describes the implemented attack framework for attack data generation, described in Section 4.6.9

4.6.5 Simulation scenario

Scenarios are used in building energy simulations to represent a building planned for construction, or a pre-existing building. In this case however, no building plan or pre-existing building exists, and thus to derive values for a generic building, a scenario was defined. The scenario selected was to represent a typical office building's HVAC system in Perth, Australia. Building types are classed based on their purpose, with design guidelines for construction materials, air changes and occupancy levels defined by the National Construction Code of Australia (NCC) (NCC, 2016). An office building is denoted as a Class 5 building in Australia, with many variables for the scenario taken from the NCC Building Code of Australia Volume one and best practices guidelines (NCC, 2016). A core requirement for the scenario is the size of the building, from which physical simulation calculations are derived. The details of the average office building sizes in Australia were retrieved from the commercial buildings baseline study, undertaken by the Australian government Department of the Environment and Energy (Phillips, 2014). From the provided dataset, Perth was selected for the average floor space containing both real and predicted data. Given the research aim is cyber security of BAS, not architecture or building energy simulation, the scenario building defined is basic, and representative of the minimum requirements for data generation, and subsequent network traffic simulation. Similarly, no year is defined for the simulation, thus the average floor space for a Class 5 office in Perth of $2912m^2$ is used as a comparative measure. The simulation is designed in such a way, that each device is encapsulated in a virtual machine, therefore there were some limitations in regards to the size of the scenario defined. The floor space is relatively large due to the multi-tenant office buildings in the Perth CBD. When designing a simulation to represent an office space this large, the number of virtual machines required were larger than could be accommodated with the hardware capabilities at the researcher's disposal. As such, the scenario size was reduced to be a smaller subset representing a business which encompases $480m^2$, consisting of both offices and computer laboratories. From the defined sizing and best practices, the Australian HVAC standards defined by the Australian Institute of Refrigeration, Air Conditioning and Heating (AIRAH) were applied to detail the devices to be used in the scenario (AIRAH, 2015a, 2015b). There are three air handling units, each supplying cooling to one office zone, sized $120m^2$ and one computer laboratory zone, sized $40m^2$ See Figure 4.12 for the defined HVAC system scenario layout, which is based on a default Variable air volume (VAV) system defined in ASHRAE (2009)[pp19.23]. The scenario simulates the operation of a BACnet/IP managed BAS for the month of January, which is summer in the southern hemisphere.

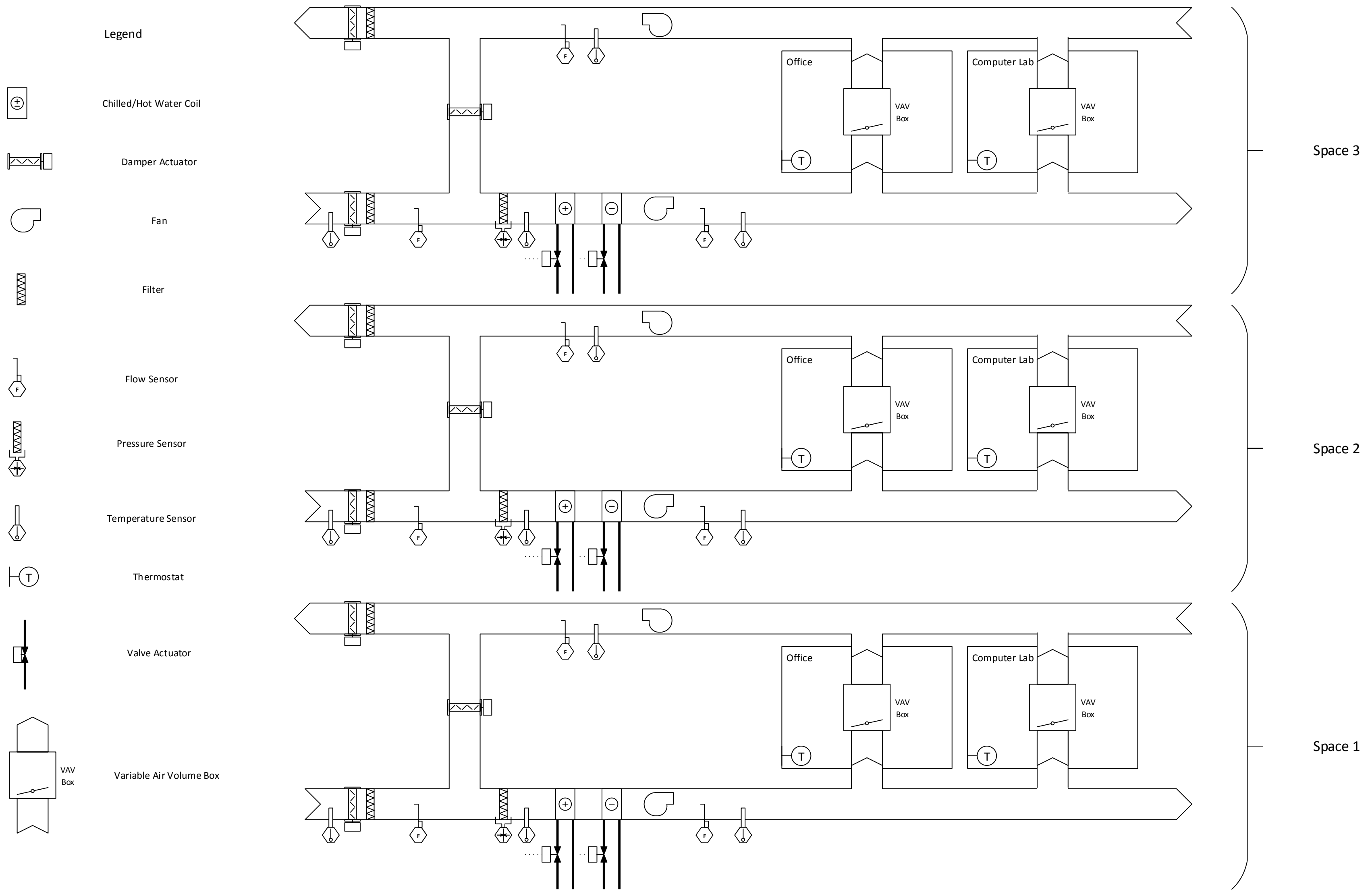


Figure 4.12: Building HVAC plan for defined scenario

4.6.6 Physical level simulation

There are a range of devices which are controlled and monitored by a BAS. As part of a cyber-physical system sensors measure various environmental phenomena. These values are then sent to controlling devices over the network, where controlling devices use predefined logic to enact change on the system. In HVAC engineering and building energy simulation, a range of equations are used to describe the monitored phenomena. For the defined scenario, and subsequent devices acting in the system, a range of data were identified to be generated for operation of the simulation. The devices and data generated are detailed in Table 4.14.

Table 4.14: Identified devices to be modelled for the simulation

Device	Generated Data
Temperature Sensor (External)	External Temperature
Temperature Sensor (Pre-Coil)	Pre-Coil Temperature
Temperature Sensor (Return Air)	ReturnAir Temperature
Zone Thermostat	Zone 1 Temperature
Zone VAV box	Zone 1 Airflow Volume
Zone VAV Box	Zone 1 Pressure
Zone VAV Box	Zone 1 Damper Settings
Zone Thermostat	Zone 2 Temperature
Zone VAV box	Zone 2 Airflow Volume
Zone VAV Box	Zone 2 Pressure
Zone VAV Box	Zone 2 Damper Settings
Flow Sensor	Supply Fan Air Volume
Supply Fan	Supply Fan Speed
Flow Sensor	Return Fan Air Volume
Return Fan	Return Fan Speed
Damper Actuator	Recycle Air Damper Settings
Damper Actuator	Exhaust Air Damper Settings
Damper Actuator	Supply Air Damper Settings
Valve Actuator	Cooling Valve Actuator Settings
Valve Actuator	Heating Valve Actuator Settings
Flow Sensor	Intake Airflow Sensor
Pressure Sensor	Pre-Coil Pressure

The majority of devices in the network are dependent on other devices for reporting data and for control of the system to provide optimum performance. The relationships between these devices can be described as control loops, of which the major control loops are the air handling unit, chiller control and zone control. The air handling unit control loop defines how much air will be ejected, recycled and taken into the system, based on the temperature of the air returning from the zones, the external temperature, the minimum required fresh air flow rate, and the amount of air required to cool the zones. The chiller control loop defines the degree to which the valves providing cooling into the system are open. The degree is determined by the Pre-coil temperature, and the cooling load of the system. In the final control loop the zone control describes how much cooled air is to be provided to the zone based on the internal temperature of the zone. The thermostat in the zone thus controls the zone damper to limit or increase air flow into the zone, which affects the resistance to air in the system. With altered resistance the pressure changes which requires action from the supply fan to maintain a static pressure setpoint.

The relationship and impact of external and internal temperature on a building is complicated, and is the focus of much research in the building simulation domain (ASHRAE, 2009). HVAC systems are defined by two core concepts, the cooling load, which defines the amount of cooling required to keep a building at a set point temperature, and the heat gain, which is the rate at which heat is generated, or conveyed into a building. ASHRAE defines a range of heat gains for a building classified as either external or internal gains. External gains are derived from the impact of solar radiation on the construction materials such as walls, roofs and windows. Internal gains are derived from the impact of occupants, lighting, and equipment emitting heat (ASHRAE, 2009). Models exist for calculating both external and internal gains, based on relevant international and national standards including the NCC, AIRAH, and ASHRAE, models for internal heat gains range from simplistic hourly operational models to models derived from real building occupancy sensor, equipment power drain and light/heat emission data. For the defined scenario, the schedules for occupancy, equipment and lighting usage are derived from Specification JV of the NCC standard, detailed in Table 4.15, whereby the schedules detail the expected percentage of maximum operation percentage at a given time. The heat gain implied by these profiles are calculated based on values provided by the NCC standard, NCC (2016, p402,p446), based on the class of room in operation, which is 75W sensible gain per person, 50W latent heat gain per person, 9W gain for lights and 15W for equipment. Equation 4.5 is used to calculate the equipment and lighting heat gains, Equation 4.6 is used to calculate the occupant sensible and latent heat gains.

$$
\begin{aligned}
&H_t = CAS_t \\
&\text{where} \\
&H_t = \text{heat gain at time } t \\
&C = \text{Energy use in Watts based on space class} \\
&A = \text{Area of space} \\
&S_t = \text{Schedule percentage at time } t
\end{aligned}
\qquad \text{(Equation 4.5)}
$$

$$
\begin{aligned}
&H_t = O(\frac{A}{P})S_t \\
&\text{where} \\
&H_t = \text{heat gain at time } t \\
&O = \text{Energy use in Watts based on occupancy per } m^2 \\
&A = \text{Area of space} \\
&P = \text{Area per Person in } m^2 \\
&S_t = \text{Schedule percentage at time } t
\end{aligned}
\qquad \text{(Equation 4.6)}
$$

Table 4.15: Occupancy, Equipment and Light schedules used in the scenario, adapted from Table 2b of NCC (2016, p394)

Time Period	Occupancy		Lighting		Equipment		AirConditioning	
	Weekday	Weekend	Weekday	Weekend	Weekday	Weekend	Weekday	Weekend
0:00-1:00	0%	0%	10%	10%	10%	10%	Off	Off
1:00-2:00	0%	0%	10%	10%	10%	10%	Off	Off
2:00-3:00	0%	0%	10%	10%	10%	10%	Off	Off
3:00-4:00	0%	0%	10%	10%	10%	10%	Off	Off
4:00-5:00	0%	0%	10%	10%	10%	10%	Off	Off
5:00-6:00	0%	0%	10%	10%	10%	10%	Off	Off
6:00-7:00	0%	0%	10%	10%	10%	10%	Off	Off
7:00-8:00	15%	0%	40%	10%	25%	10%	On	Off
8:00-9:00	60%	0%	80%	10%	70%	10%	On	Off
9:00-10:00	100%	0%	100%	10%	100%	10%	On	Off
10:00-11:00	100%	0%	100%	10%	100%	10%	On	Off
11:00-12:00	100%	0%	100%	10%	100%	10%	On	Off
12:00-13:00	100%	0%	100%	10%	100%	10%	On	Off
13:00-14:00	100%	0%	100%	10%	100%	10%	On	Off
14:00-15:00	100%	0%	100%	10%	100%	10%	On	Off
15:00-16:00	100%	0%	100%	10%	100%	10%	On	Off
16:00-17:00	100%	0%	100%	10%	100%	10%	On	Off
17:00-18:00	50%	0%	80%	10%	60%	10%	On	Off
18:00-19:00	15%	0%	60%	10%	25%	10%	Off	Off
19:00-20:00	5%	0%	40%	10%	15%	10%	Off	Off
20:00-21:00	5%	0%	20%	10%	15%	10%	Off	Off
21:00-22:00	0%	0%	10%	10%	10%	10%	Off	Off
22:00-23:00	0%	0%	10%	10%	10%	10%	Off	Off
23:00-24:00	0%	0%	10%	10%	10%	10%	Off	Off

Naturally, external temperature values form a cyclical pattern based on the time of day; which impacts the internal temperature of a building. While the relationship is not linear, it is often correlated (Degelman, 2004). When the temperature is high outside, the cooling required to maintain an internal setpoint is increased. When the temperature is lower, less cooling is required, with some control strategies switching off to rely on natural cooling to save energy. The optimal zone of operation, whereby the HVAC can regain energy, is called the deadband, which is set as a one degree distribution from the setpoint. Further, a proportional band is defined, which is set as one degree below the deadband for heating and two degrees above the deadband for cooling. The proportional band defines the ramp time for heating or cooling from the HVAC. For this scenario, the setpoint is set as the Australian standard for Perth of 23 Degrees Celsius AIRAH (2015a), the control bands are detailed as Figure 4.13.

Modelling external heat gain is more complex, due to environmental phenomena and the various thermal properties of construction materials. External heat gains use cooling factors and the properties of construction material for defining the absorbance of the building. Further, long-term weather data are used to calculate the heat gain. While the impact of reducing thermal load from construction materials is important for building simulations relating to energy usage and air-

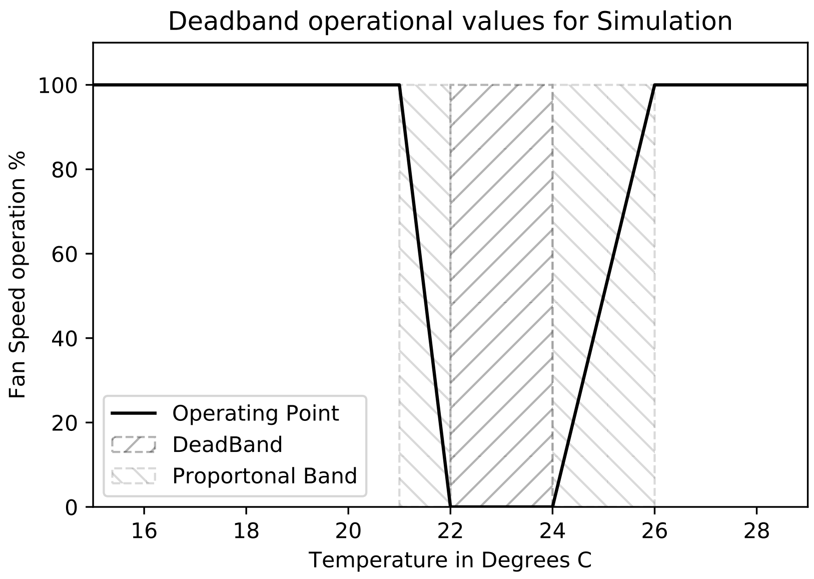


Figure 4.13: Control bands and setpoint for scenario, defined by NCC (2016)

conditioning design, for the use in this research it provides added complexity to the calculations which have an undetermined impact on the variance of variable values. As such, an assumption-based approach has been used for determining the effect of solar radiation on the zone temperatures as 10% of external temperature interacts with the internal zone temperature.

The cooling load is the total amount of energy a cooling system is required to produce to maintain a specific setpoint temperature ASHRAE (2009, p18.1). The peak cooling load is derived to select the appropriate equipment to use in a HVAC system. There are various methods for deriving the cooling load defined by ASHRAE and AIRAH, with the ASHRAE recommended methods being the heat balance (HB) method and the radiant time series (RTS) method (ASHRAE, 2009). From the nature of these methods, there are algorithms which explain complex natural phenomena, based on the physical characteristics of a building, its location, and usage. Given the purpose of the simulation is to generate network traffic, rather than accurate physical based variables, an assumption based approach is taken to generating the cooling load. For energy modelling, typically, the cooling load is generated at a per hour rate, however, for generating network simulation data a fidelity level of seconds is required. ASHRAE states that the mean bias error for energy simulations can be up to 30% for hourly values. It is also unclear which simulation variables are of most importance for the design cooling load. For the purpose of this research, generating data at a fidelity of seconds, the values generated using the prescribed methods, HB or RTS may not be accurate. Thus to reduce the complexity of the calculations, an assumption is made with the cooling output required for an office building defined as $180W/m^2$ (Curnow & Curnow, 2014). The equation used to generate a synthetic cooling load for the scenario is shown in Equation Equation 4.7. The cooling load per floor of the system is thus 28.8kW ($\frac{160m^2 \times 180}{1000}$).

§4.6.6.1 - §4.6.6.7 describe the models used for data generation in this research.

$$\begin{aligned}
&L = \frac{AO}{1000} \text{in kW} \\
&\text{where} \\
&L = \text{Cooling load} \\
&A = \text{Total area of space} \\
&O = \text{Cooling Output constant} 180 W/m^2
\end{aligned} \qquad \text{(Equation 4.7)}$$

4.6.6.1 External temperature generation

Degelman (2004) describes a Monte Carlo method of generating weather, using both deterministic and stochastic models. The aim of the models in Degelman (2004) is the generation of synthetic weather data for simulating building thermal loads, the flexible fidelity of the data generated by the model provides the necessary level of data for the device logic designed for the network simulation. Degelman (2004) notes two systems which have strong interrelationships which affect the majority of financial impact of a building, namely, the thermal envelope of the building, and the air conditioning system. The purpose of building energy performance simulation is thus for cost benefit analysis of various operating profiles and lifecycles of a building. The most prominent mechanism of heat flow in buildings is the climate in which the building is situated, making weather data an important factor for simulation. Degelman's model has two parts, a daily deterministic model representing the cyclical diurnal pattern of temperature, and a stochastic model, which uses the Monte Carlo method for deriving a sequence of days from the normalised cumulative distribution function of real data recordings.

The model was applied to generate external temperature values for the simulated scenario. Correlated, half-hourly weather readings from the Perth meteorological weather station (station number 9021) were retrieved from Peterson (2013) for each day in January for the period 1998 to 2013. The external temperature readings were used to derive a range of descriptive statistics on a per day basis.

The deterministic model used requires the sunrise time and solar noon, referred to as zenith time henceforth. The typical values for January in Perth were retrieved from 'Perth, Western Australia, Australia - Sunrise, Sunset, and Daylength' (2017) and an assumption was made to hold the sunrise and zenith values constant on the hour for the month, with sunrise at 5am and zenith at 12pm. Three algorithms, Equation 4.8, Equation 4.9, Equation 4.10 were used to calculate the temperatures for a day in the deterministic model. The stochastic model generates the maximum temperature and minimum temperature for each day, which is then chained together to form a simulated month of data.

$$T_t = T_{ave0} - \frac{\Delta T}{2}\cos[\pi\frac{(t-t_R)}{(Z-t_R)}]$$

where

$T_t =$ temperature at time t

$t_{ave0} =$ average morning temperature (Equation 4.8)

$\Delta T =$ temperature range

$t_R =$ sunrise time

$Z =$ zenith time

$$T_t = T_{ave1} + \frac{\Delta T\prime}{2}\cos[\pi\frac{(t-Z)}{t_R\prime + N}]$$

where

$T_{ave} =$ the average evening temperature (Equation 4.9)

$\Delta T\prime =$ the evening temperature range $(T_{max} - T_{min}\prime)$

$t_R\prime =$ the time of sunrise the next day

$N =$ constant

$$T_t = T_{ave1} + \frac{\Delta T\prime}{2}\cos[\pi\frac{t+N}{T_R\prime + N}]$$

where

$T_{ave} =$ the average evening temperature (Equation 4.10)

$\Delta T\prime =$ the evening temperature range $(T_{max} - T_{min}\prime)$

$t_R\prime =$ the time of sunrise the next day

$N =$ constant

The probability density function (PDF) for dry-bulb temperatures almost always follow the normal distribution curve (Degelman, 2004). The objective of the stochastic model is thus to sample a mean which follows the PDF. For simulation purposes the integral cumulative density function (CDF) is used, defined as the area under the PDF curve from left to right. Thus, the lowest temperature will be positioned at the left of the CDF, and the highest temperature at the right. To derive days based on the standard deviation of real temperatures over a period, the CDF Y axis is manipulated to represent days in the month, rather than probability of occurrence, with the X axis being the temperature minus the mean temperature. The result is 31 unique normalised values from the mean, which represents each day of the month. By selecting each day from the

CDF without repetition, the pattern of the CDF and therefore the PDF of actual temperature occurrences can be replicated for the simulation. The values required for calculation are the mean temperature for each day, and the standard deviation of daily temperature. The standard deviation of daily temperatures is calculated over a range of years, following Equation 4.11.

$$\sigma = \sqrt{\frac{\sum x_i^2 - n\bar{x}^2}{n-1}}$$

where

$\sigma =$ the standard deviation of the period

$n =$ number of days in sample x_i

$\bar{x} =$ the mean for the period studied $(\sum \frac{x_i}{n})$

(Equation 4.11)

The normalised deviations are then sampled using the Monte Carlo method with a seed value for future repeatability, resulting in the monthly mean temperature pattern in Figure 4.14. To generate minimum and maximum values corresponding to the simulated monthly pattern, Equation 4.12 and Equation 4.13 were used.

$$Min_d = \bar{X}_d - \sigma \left|\hat{\bar{X}}\right|$$

where

$\bar{X}_d =$ Mean on day d

$\left|\hat{\bar{X}}\right| =$ absolute value of normalised mean

(Equation 4.12)

$$Max_d = \bar{X}_d + \sigma \left|\hat{\bar{X}}\right|$$

where

$\bar{X}_d =$ Mean on day d

$\left|\hat{\bar{X}}\right| =$ absolute value of normalised mean

(Equation 4.13)

The minimum and maximum values generated using the stochastic model are used in the deterministic model to generate temperature values for every second of the simulated January, shown as Figure 4.15. With these external values, the logic of the BAS devices can then call values at a fidelity level of seconds.

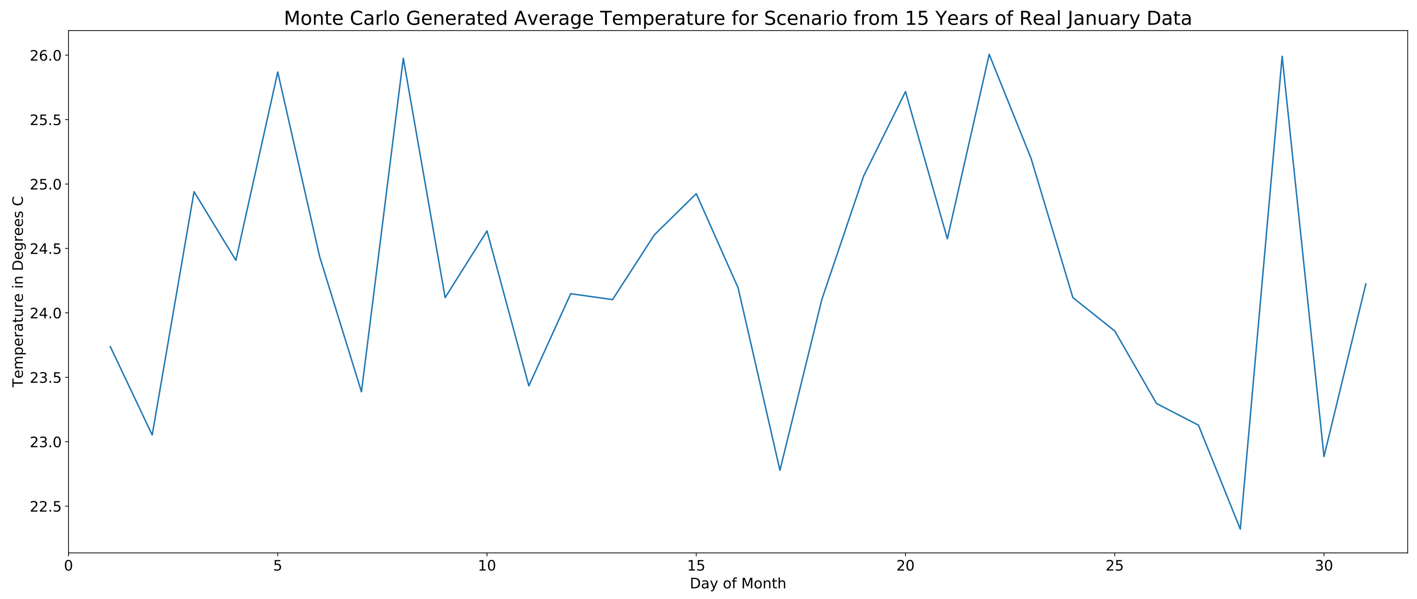


Figure 4.14: Daily average temperatures generated using the Monte Carlo algorithm

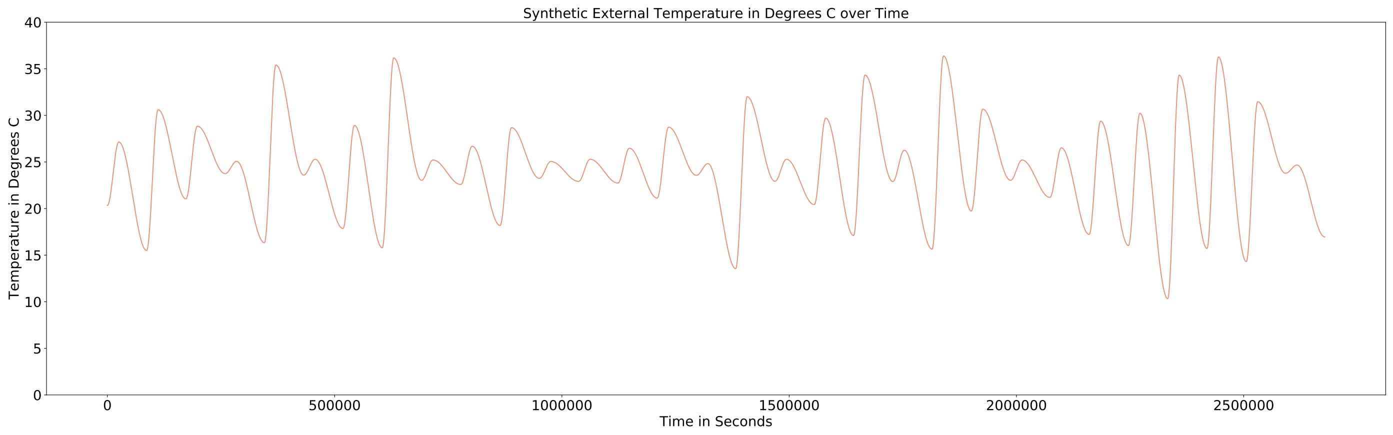


Figure 4.15: Graph of simulated January external temperature values over time

4.6.6.2 Internal zone temperature

There are two points at which the internal room temperature is measured, representing the two types of zone being simulated, namely, Office Zone and Lab Zone. As discussed in §4.6.6, there are a range of factors which affect the internal temperature of a space. A model is required for each factor, which can then be used to generate the internal zone temperature. Each zone type will have differing heat gains, due to the differences in area, and area per person specified by (NCC, 2016). The output of heat gain calculations is the power of the system used to convert this power to a temperature effect for which Equation 4.14 and Equation 4.15 are used. The internal heat gains, and subsequent temperature increases for the Office and Lab zones, are detailed in Figure 4.16

$$\frac{\frac{PT}{m}}{cp}$$

where

$P =$ Power in kW

$T =$ Time in seconds

$m =$ Mass of air in kg

$cp =$ Specifc heat of air $1.005kJ/kgC$

(Equation 4.14)

$$m = \frac{V}{\rho}$$

where

$m =$ Mass of air in kg

$V =$ Volume of air in space in kg

$\rho =$ Density of Air $1.200kg/M^3$

(Equation 4.15)

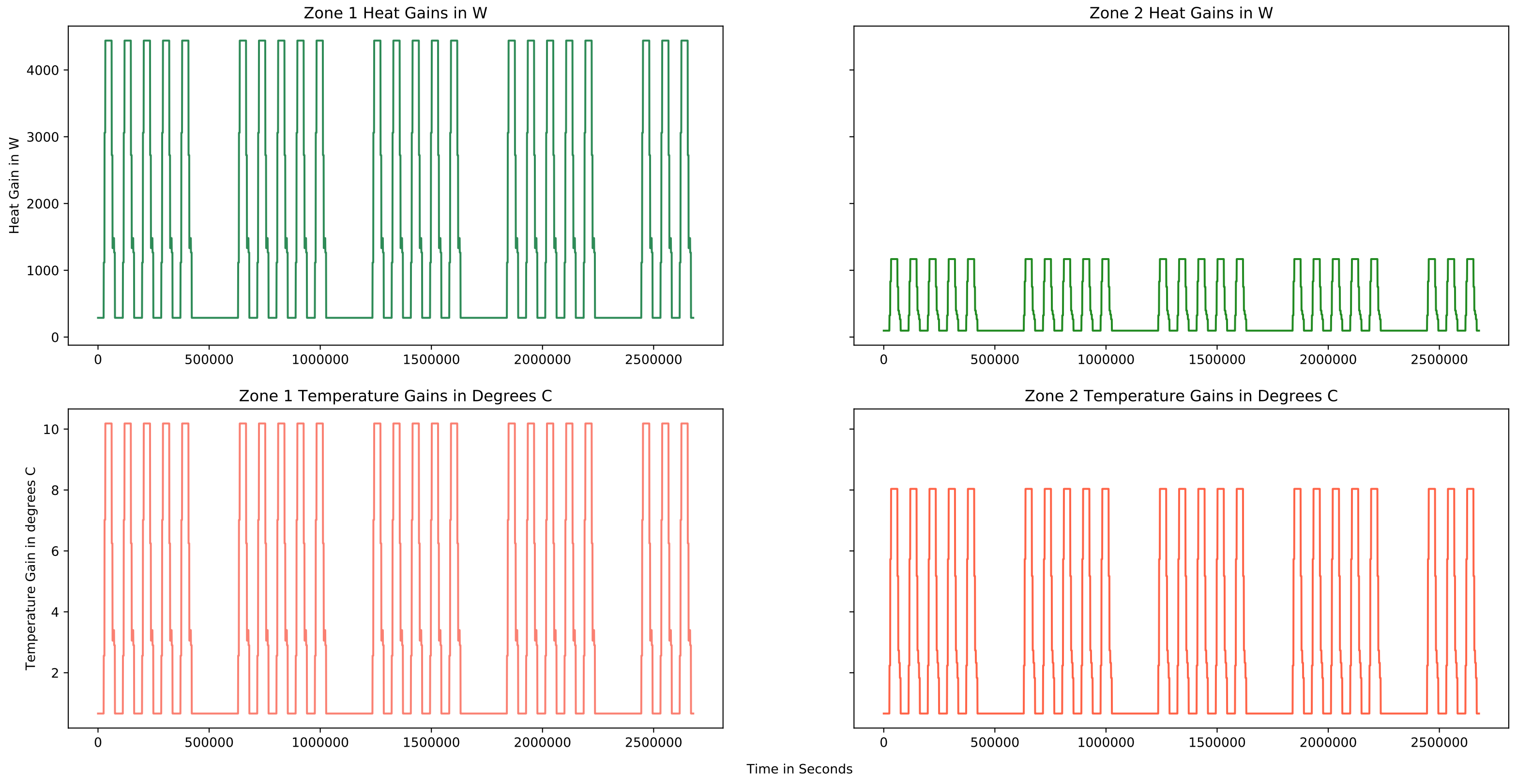


Figure 4.16: Internal heat gains and Temperature gains for the office (left) and lab (right) zones

To determine the external gain the assumption that 10% of external temperature is the external heat gain was made. The temperature effect of the internal and external gains are combined, per zone, to form the impact of all heat gains to each zone. Given the external temperature values, the power of the external temperature is derived via Equation 4.16

$$\begin{aligned} &G = TcpV\rho \\ &\text{where} \\ &G = \text{Gain in } kW \\ &T = \text{Temperature in degrees C} \\ &cp = \text{Specific heat of air } 1.005kJ/kgC \\ &V = \text{Volume of zone in } m^3 \\ &\rho = \text{Density of air } 1.205kg/m^3 \end{aligned} \qquad \text{(Equation 4.16)}$$

Given the air makeup in the zone is a mix between the supplied vent air and the existing room air, the internal temperature is generated using the following process. First, a potential temperature is generated, using Equation 4.17, a constant supply air temperature of 13 Degrees (the standard for Australia (AIRAH, 2015b)), and the previous room temperature with the current time slice heat gains.

$$\begin{aligned} &MA_t = \frac{(SA_tSAP_t) + (RA_t(100 - SAP_t))}{100} \\ &\text{where} \\ &MA_t = \text{Mixed air temperature in degrees Celsius at time } t \\ &SA_t = \text{Supply air temperature in degrees Celsius at time } t \\ &SAP_t = \text{Percentage of supply air at time } t \\ &RA_t = \text{Room air temperature in degrees Celsius at time } t \end{aligned} \qquad \text{(Equation 4.17)}$$

A control function is used, which increases or decreases the supply air percentage based on the ΔT between the setpoint temperature, and the generated potential temperature, resulting in the selected percentage of air for the next temperature generation. The actual temperature is then generated using Equation 4.17. Of the generated values, less than 70 values fall outside the range 20-26 degrees C, specified by AIRAH as acceptable temperature in an office in Australia (AIRAH, 2015a). A model incorporating control functions, such as ramp up and purge could account for the sudden increase in load at the start of each work day. Alternatively, a model which uses a data driven, realistic internal load schedule, rather than the NCC standard, would have reduced load spikes.

4.6.6.3 Damper actuators

There exist two types of damper actuators in the system, those for control of air flow in the system, termed Intake, Exhaust and Recycle, and the zone dampers controlled by the Thermostat. The Intake, Exhaust and Recycle damper actuators form an integral part of the HVAC system. The Intake damper defines the volume of air entering the building, equally, the Exhaust damper defines the volume of air leaving the system. These dampers are intrinsically linked, as the equal volume of air entering the system must also leave the system, while ignoring exfiltration loss through the building envelope. The Recycle damper allows for part of the return air stream to re-enter the HVAC process, reducing the amount of cooling and conditioning required. The total air makeup of the system is controlled by the percent of air being recycled, given the air exiting the system through the exhaust is equal to the air entering the system through the intake. Damper settings are generally static setpoints, rather than having a high degree of control, to allow for system curves to be generated for resistance in the air flow. In this system, the percentage the damper is open has a linear relation to the percentage of air entering or exiting the system. A control function was defined based on the difference in temperature between the return air temperature provided by the recycle damper and the external temperature provided by air from the intake damper. The function is detailed in Algorithm 1 and Algorithm 2.

The zone dampers are controlled by the zone thermostat. As the sensed temperature reaches its defined setpoint the dampers close, which increases the resistance in the duct, which in turn increases the fan speed to account for the pressure change. As the sensed temperature deviates from the defined setpoint, the opposite control function occurs, the dampers open, which reduces the resistance in the duct, which in turn reduces the fan speed to account for the pressure reduction. Three zone damper positions are defined for the scenario, 10%, 50% and 100%. The current damper position is determined based on the control function in Algorithm 3.

Algorithm 1: Exhaust and Intake Air Percentage Control Function

```
1 function CalculateAirPercent($MixedAirPercent$)
  Input  : Mixed air percentage
  Output: Exhaust air percentage, intake air percentage
2 $ExhaustAirPercent \leftarrow 100 - MixedAirPercent$
3 $IntakeAirPercent \leftarrow ExhaustAirPercent$
4 return $ExhaustAirPercent, IntakeAirPercent$
```

4.6.6.4 Duct static pressure sensors

There are three pressure sensors in the system, one in each zone Variable Air Volume (VAV) box, and one standalone sensor before the heating and cooling coils. Air flows from high pressure to low pressure, thus HVAC systems are designed to have lower pressure in areas which air is to flow to, such as zones. Static pressure is impacted by two major factors. The friction caused by the duct

Algorithm 2: Damper Percentage Control Function

```
1  function DamperControlFunction($ExternalTemperature, ReturnAirTemperature$)
   Input  : Two temperature values, $ExternalTemperature$ and $ReturnAirTemperature$
   Output: Mixed air percentage, recycle damper percentage, exhaust damper percentage, intake
           damper percentage
2  $DT \leftarrow ExternalTemperature - ReturnAirTemperature$
3  if $-3 < DT < 3$ then
4      $MixedAirPercent \leftarrow 50$
       $ExhaustAirPercent, IntakeAirPercent \leftarrow CalculateAirPercent(MixedAirPercent)$
5      return $MixedAirPercent, ExhaustAirPercent, IntakeAirPercent$
6  else if $DT <= -3$ then
7      $MixedAirPercent \leftarrow 20$
       $ExhaustAirPercent, IntakeAirPercent \leftarrow CalculateAirPercent(MixedAirPercent)$
8      return $MixedAirPercent, ExhaustAirPercent, IntakeAirPercent$
9  else if $DT >= 3$ then
10     $MixedAirPercent \leftarrow 70$
       $ExhaustAirPercent, IntakeAirPercent \leftarrow CalculateAirPercent(MixedAirPercent)$
11     return $MixedAirPercent, ExhaustAirPercent, IntakeAirPercent$
12 $RecycleDamper, ExhaustDamper, IntakeDamper \leftarrow$
   $MixedAirPercent, ExhaustAirPercent, IntakeAirPercent$
13 return $RecycleDamper, ExhaustDamper, IntakeDamper$
```

Algorithm 3: Zone Damper Percentage Control Function

```
1  function ZoneDamperControlFunction($ZoneTemperature$)
   Input  : One temperature value, $ZoneTemperature$
   Output: Zone damper position
2  if $ZoneTemperature < 21.0$ then
3      $ZoneDamperPercent \leftarrow 100$
4      return $ZoneDamperPercent$
5  else if $ZoneTemperature < 22.5$ then
6      $ZoneDamperPercent \leftarrow 50$
7      return $ZoneDamperPercent$
8  else if $ZoneTemperature < 23.5$ then
9      $ZoneDamperPercent \leftarrow 10$
10     return $ZoneDamperPercent$
11 else if $ZoneTemperature <= 25.0$ then
12     $ZoneDamperPercent \leftarrow 50$
13     return $ZoneDamperPercent$
14 else if $ZoneTemperature > 25.0$ then
15     $ZoneDamperPercent \leftarrow 100$
16     return $ZoneDamperPercent$
```

material, air filters and dampers over a length of ducting, and any deviations in ducting size or curves in the duct. Fans introduce static pressure to the air stream, allowing the air to flow from the fan to the zone areas. The fan's aim is to maintain a constant static pressure at the point of air proliferation to the zone. When a damper moves, the resistance in the system increases which in turn changes the pressure in the system. The fan must account for this pressure change, and increase or decrease fan speed to add the appropriate amount of pressure into the system. Given dampers have set positions, the resistance added, and thus pressure changes can be described. The pressure changes in a system can be determined from one pressure reading coupled with a volumetric air flow reading using the second Affinity law, see Equation 4.18.

$$SP_2 = SP_1(\frac{Q_2}{Q_1})^2 \qquad \text{(Equation 4.18)}$$

where

$SP_2 =$ The pressure at air flow reading Q_2 in Pascals

$SP_1 =$ The pressure at air flow reading Q_1 in Pascals

$Q_2 =$ Next volumetric air flow in M^3/s

$Q_1 =$ Current volumetric air flow in M^3/s

Given the first pressure reading for each damper position, all pressure readings for the air flows entering each zone can be calculated. When the system changes resistance due to a damper moving, the pressure curve generated by Equation 4.18 moves, thus Equation 4.18 is seeded with each zone's assigned static pressure point. To generate the complete zone static pressure, the static pressure from each curve is selected based on the current damper position using Algorithm 4, accounting for the pressure loss caused by duct length. The pre-coil pressure sensor values are also generated using the second affinity law from Equation 4.18, with a lower seed value to represent the lower pressure value before the fan imparts additional pressure into the airstream.

4.6.6.5 Fans

There are two fans in the scenario system, one for supplying air the zones, and one for returning air from the zones. Each has equal actions to allow for balancing of air supply and return, thus the fan speed and static pressure from each fan is identical. When one fan speed is known, all future fan speeds can be calculated using the first affinity law, see Equation 4.19. The fan speed of the supply fan is calculated using the total supply air derived from the sum of both zones generated air flows. The static pressure introduced into the airstream by the fan is calculated using the second affinity law, replacing the air flow with the fan speed, see Equation 4.20. The seed fan speed was

Algorithm 4: Zone static pressure sensor data generation

```
Input  : Three lists of generated static pressure values, one list of the zone damper positions, one
         list of supply air volume and the duct length to the zone,
         $ZonePressureDamper100, ZonePressureDamper50,$
         $ZonePressureDamper10, ZoneDamperPercent, ZoneSupplyAirVolume, DuctLength$
Output: Zone pressure sensor readings
1  $Control \leftarrow 10$
2  $Loss \leftarrow DuctLength * Control$
3  Set $ZonePressure$ to []
4  for $i \leftarrow 0$ to $len(ZoneSupplyAirVolume)$ do
5      if $ZoneDamperPercent[i] == 100$ then
6          $ZonePressure[i] \leftarrow ZonePressureDamper100[i] - Loss$
7      else if $ZoneDamperPercent[i] == 50$ then
8          $ZonePressure[i] \leftarrow ZonePressureDamper50[i] - Loss$
9      else if $ZoneDamperPercent[i] == 10$ then
10         $ZonePressure[i] \leftarrow ZonePressureDamper10[i] - Loss$
11 end
12 return $ZonePressure$
```

set to 700RPM, the seed fan static pressure was set to 500 Pa.

$$RPM_2 = RPM_1(\frac{Q_2}{Q_1})$$

where

RPM_2 = Next fan speed in Revolutions Per Minute (Equation 4.19)

RPM_1 = Current fan speed in Revolutions Per Minute

Q_2 = Next volumetric air flow in M^3/s

Q_1 = Current volumetric air flow in M^3/s

$$SP_2 = SP_1(\frac{RPM_2}{RPM_1})^2$$

where

SP_2 = The pressure at air flow reading Q_2 in Pascals (Equation 4.20)

SP_1 = The pressure at air flow reading Q_1 in Pascals

RPM_2 = Next fan speed in Revolutions Per Minute

RPM_1 = Current fan speed in Revolutions Per Minute

4.6.6.6 Air flow

For this research, air flow is assumed to be turbulent flow, thus with a constant velocity per unit time, inline with the assumptions posited by ASHRAE (2012a). Air flow is sensed at five points in the system scenario, namely, the intake air flow, supply air flow, zone air flows and return air flow. For each air flow, the supply air mass flow is calculated, using Equation 4.21. From the mass flow, the volumetric flow is calculated using Equation 4.22. The supply air flow is the sum of both air flows entering the zone. Similarly, the return air flow is the sum of the zone air flows leaving the zone. The Intake air flow volume is generated based on the percentage of external air flowing into the system, defined by the damper percentage. Thus, when the Temperature is cooler outside, more air is pulled into the system due to the damper control functions. As air flow before and after the fan must be equal, the Intake air flow is generated using Equation 4.23.

$$m = \frac{H}{cp\Delta(T_r - T_s)}$$

where

$m =$ The mass flow rate in kg/s

$H =$ The sensible heat gain in kW (Equation 4.21)

$cp =$ Specific heat of air $1.005kJ/kgC$

$T_r =$ Room temperature in Degrees C

$T_s =$ Supply temperature in Degrees C

$$V = \frac{m}{\rho}$$

where

$V =$ Volume flow rate in m^3/s (Equation 4.22)

$m =$ The mass flow rate in kg/s

$\rho =$ Density of air $1.205kg/m^3$

$$Q_i = I_p(Q_t)$$

where

$Q_i =$ Intake air flow in M^3/s (Equation 4.23)

$I_p =$ Intake damper percentage

$Q_t =$ Total system air flow in M^3/s

There are three different internal points at which temperature is recorded in the ventilation system, defined as; Pre-coil, Post-coil, and Return-vent. The Pre-coil temperature is a mix of the return temperature and the temperature of the external fresh air. The percentage of air mix is determined by the Intake and Recycle dampers detailed in §4.6.6.3. The temperature value generated determines the cooling valve uptime for supply air cooling, detailed in Equation 4.24.

$$
\begin{aligned}
&MA_t = \frac{(OA_t OAP_t) + (RA_t(100 - OAP_t))}{100} \\
&\text{where} \\
&MA_t = \text{Mixed Air temperature in degrees Celsius at time } t \\
&OA_t = \text{Outside air temperature in degrees Celsius at time } t \\
&OAP_t = \text{Percentage of outside air at time } t \\
&RA_t = \text{Return air temperature in degrees Celsius at time } t
\end{aligned}
\qquad \text{(Equation 4.24)}
$$

The Post-coil temperature is the reading generated from the Pre-coil temperature passing across the cooling and heating coils to reach a setpoint temperature, which is then supplied to the zones. In Australia, the recommended supply temperature is 12-13 Degrees C (AIRAH, 2015b), as such, a constant value of 13 Degrees C is used as the Post-coil temperature.

The Return-vent temperature is generated using the mixed air temperature Equation 4.24. The air flow supplied to the zone is equal to the air flow exiting the zone into the return duct. As such, the total amount of return air is the sum of the Office zone and Lab zone supply air. The percentage of Office return and Lab return airs are calculated, and used in Equation 4.24 to generate the return air temperature.

4.6.6.7 Valves

There are two valves with attached actuators in the scenario, controlling the heating coil valve and cooling coil valve respectively. For the summer profile in this scenario, the heating coil is always fully closed, thus the valve percentage is a constant value of 0%. The cooling coil is calculated using a number of equations. First, the mass flow rate of the coolant in the pipe is calculated, see Equation 4.25. Second, the sensible cooling of the chiller is calculated. As per ASHRAE (2009), with the assumption that no heat is lost through the pipe, the calculation for the sensible cooling of the chiller is equal to the heat removed from the air, and the heat absorbed by the coolant flow. This value is calculated using Equation 4.26 taken from ASHRAE (2009)[p23.7 e2a, e2b]. Next, the coolant output temperature is calculated using Equation 4.27, which also results in the maximum heat absorbed by the coolant. Finally, the maximum heat absorbed is taken as a constant, solving

Equation 4.28 to determine the valve percentage for the cooling coil valve.

$\dot{m} = \rho V A$

where

$\dot{m} =$ Mass flow rate in kg/s

$\rho =$ Density of water in kg/m^3 (Equation 4.25)

$V =$ Velocity of water inside pipe in l/s

$A =$ Cross-sectional area of pipe in m^2

$Q = 1000 W_a cp \Delta T$

where

$Q =$ Cooling capacity in W

$W_a = 60 \rho A_a V_a$

$cp =$ Specific heat of air in kJ/kg (Equation 4.26)

$\Delta T =$ Intake Air Temperature - Output Air Temperature

$\rho =$ Density of Air in kg/m^3

$A_a =$ Coil Face in m^2

$V_a = \frac{m_a ir}{A_a}$ in kg/s

$C_o = C_i + \frac{Q}{1000 \dot{m} cr}$

where

$C_o =$ Coolant Output in Degrees C (Equation 4.27)

$C_i =$ Coolant Intput in Degrees C

$\dot{m} =$ Mass flow rate in kg/s

$cr =$ Specific heat of water in Degrees C

$VP_t = \frac{\Delta T}{-C}$

where

$VP_t =$ Valve percentage at Time t (Equation 4.28)

$\Delta T = C_o - C_i$ Air Temperature in Degrees C

$-C = \frac{Q}{1000 \dot{m} cr}$ in Degrees C

4.6.7 Device level simulation

For simulating a BACnetwork, appropriate digital representation of each device is required. In addition to the devices which generate values, described in §4.6.6, a controller device is required which controls and monitors the damper actuators operating in the air handling unit (AHU), in addition to a human machine interface (HMI) device to view trends in the system. The BACnet standard defines six device profiles in Annex L (SSPC-135, 2012), which defines the minimum set of tasks a device must perform to be classed as a specific type of device. Analysis of the required devices for the simulation reveal five types of profiles in use, detailed in Table 4.16.

Table 4.16: Simulation devices classified via device profile

Device	Device Profile
AHU Controller	Advanced Application Specific Controller
Zone1 Thermostat	Application Specific Controller
Zone2 Thermostat	Application Specific Controller
HMI	Advanced Operator Workstation
Zone1 VAV box	Smart Actuator
Zone2 VAV box	Smart Actuator
Supply Fan	Smart Actuator
Return Fan	Smart Actuator
Recycle Damper Actuator	Smart Actuator
Exhaust Damper Actuator	Smart Actuator
Supply Damper Actuator	Smart Actuator
Cooling Valve Actuator	Smart Actuator
Heating Valve Actuator	Smart Actuator
Temperature Sensor (External)	Smart Sensor
Temperature Sensor (Pre-Coil)	Smart Sensor
Temperature Sensor (Return Air)	Smart Sensor
Supply Flow Sensor	Smart Sensor
Return Flow Sensor	Smart Sensor
Intake Flow Sensor	Smart Sensor
Pre-Coil Pressure Sensor	Smart Sensor

To define the devices, the analysis of extracted profiles undertaken in §4.6.7.1 was used to construct a generic device type for each profile. Each device incorporates the most prominent objects and services implemented.

4.6.7.1 Device profile extraction

Due to the object-based approach to constructing BACnet devices, each vendor implementation can differ for devices of the same type. As such, the task of defining a generic device is complicated, given there are over 1,000 registered BACnet vendors. In order for a BACnet device to be sold compliance testing is undertaken with a resulting Protocol Implementation Compliance Statement (PICS) generated for each device. The PICS details the device profile, supported network services, objects and properties of the device, differentiating between required and optional. In addition, the BACnet testing and compliance process generates a product listing, which also contains the

PICS data. ASHRAE provides a wide range of vendors PICs, and subsequent product listings BTL (2018). ASHRAE/BACnet international provides a standard template for generating a PICS, however from analysis of the retrieved files, and confirmation via Esquivel-Vargas et al. (2017), the variance in template is not restricted between vendors. Rather, the same vendor may have multiple representations of a PICS between different devices. The common datatype between all PICS was the file format of Portable Document Format (PDF). Thus, to extract the PICS data in a programmatic form, the approach outlined in Esquivel-Vargas et al. (2017) was investigated, namely, transformation of a PDF into a data structure which can be manipulated using string comparison algorithms. After analysis of the data extraction processes for PICS, the product listings were investigated which contain the same information on devices as the vendor generated PICS, however, the product listings are in a more structured format. The product listing PDFs used tables to represent the core information about each device, thus the PDFs were converted to *html* format to utilise tags and manipulation libraries such as *Python's Beautiful Soup*. The template for data extraction was derived from the product listing, allowing for extraction of PICS data in a structured way. A total of 389 product listings were retrieved for five device profiles, namely, smart sensor, smart actuator, application specific controller, advanced application specific controller and operator workstation. The location of the objects table is dependent on the number of services the device offers, as each class of service is detailed in a separate table. As such, a template was defined for each type of device profile. Following the process in Figure 4.17, 389 files were directly converted to *html* using *Adobe Reader Pro DC 2018* (Adobe, 2018). Of these files, 365 extracted correctly using the defined template, and data extraction process. Data extraction was undertaken with the *Python* library *Beautifulsoup4*, regular expressions and the Aho-Corasick algorithm for string matching. Verification of extraction was undertaken using further regular expressions and the *Python* package pandas. The files which did not extract according the template were due to a range of discrepancies in the data, from required objects missing in the object table, to incorrect table structures. Therefore, the 24 files were discarded. The objects in use by these device types are detailed in Table 4.17.

Services were matched to devices based on the BIBB classifications, see Table 4.18. 13 non-standard defined BIBBs were encountered. In some product listings, where a BIBB by definition did not have corresponding client and server functionality, the client and server BIBB classifier, A and B respectively were omitted, resulting in further classification of seven BIBBs. Two BIBBS contained in one device listing erroneously listed the class of the BIBB, DS-TS-B/DS-UTC-B rather than DM-TS-B/DM-UTC-B respectively, as such those values were updated accordingly. The remaining four BIBBs contained the characters *T-XXX-XXX*, whose meaning could infer all BIBBS of the T class, a placeholder which was not finalised, or a misprint. As such, these four BIBB types were not converted to service functions. With the types of objects used in real devices defined, the objects, properties and services used by these identified devices were then used to construct BACnet devices for the simulation. From the derived profiles, the objects and services implemented in the *BACnet open stack* were identified. All objects used in the Actuator, Controller, Sensor and Workstation

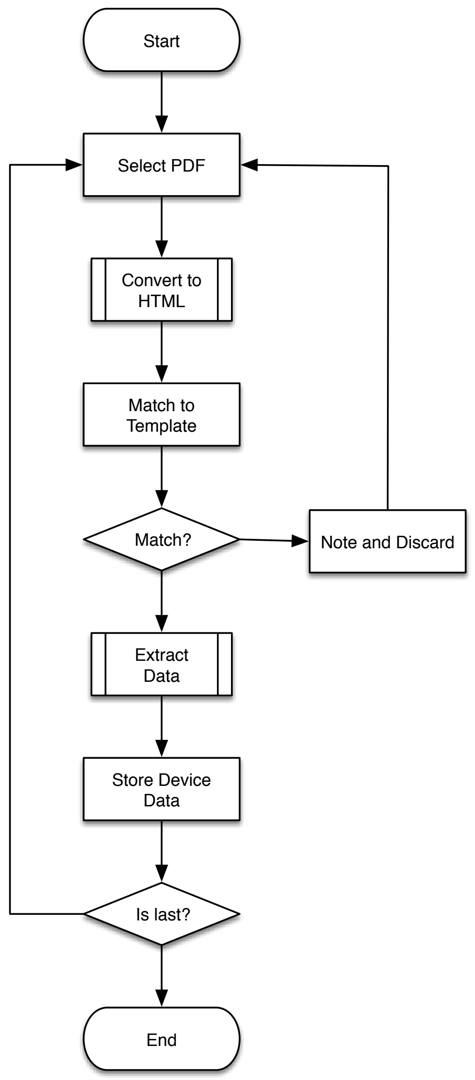


Figure 4.17: Process diagram of BACnet device product listing statement data extraction and transformation

are implemented. However, five objects from the Advanced Controller profile, namely, *Notification class*, *Calendar*, *Program*, *Loop* and *Event enrolment* were not implemented, and thus omitted.

Table 4.17: Object occurrence in device profiles retrieved from device product listing statements

Objects	Sensor	Controller	Actuator	Advance Controller	Work station	Sensor Percent	Controller Percent	Actuator Percent	Advance Controller Percent	Work station Percent
accumulator	3	2	4	3	0	21.43	0.78	23.53	5.56	0.00
alert-enrollment	0	4	0	2	0	0.00	1.56	0.00	3.70	0.00
analog-input	13	223	14	49	4	92.86	87.11	82.35	90.74	16.67
analog-output	2	133	10	44	4	14.29	51.95	58.82	81.48	16.67
analog-value	6	224	11	54	5	42.86	87.50	64.71	100.00	20.83
averaging	0	1	0	0	0	0.00	0.39	0.00	0.00	0.00
binary-input	5	195	12	45	4	35.71	76.17	70.59	83.33	16.67
binary-output	3	144	10	44	4	21.43	56.25	58.82	81.48	16.67
binary-value	3	194	11	52	5	21.43	75.78	64.71	96.30	20.83
bitstring-value	0	5	2	0	0	0.00	1.95	11.76	0.00	0.00
calendar	0	14	0	54	4	0.00	5.47	0.00	100.00	16.67
characterstring-value	0	2	0	1	0	0.00	0.78	0.00	1.85	0.00
command	0	5	0	3	0	0.00	1.95	0.00	5.56	0.00
date-value	0	1	0	0	0	0.00	0.39	0.00	0.00	0.00
device	14	256	17	54	24	100.00	100.00	100.00	100.00	100.00
event-enrollment	0	7	0	15	4	0.00	2.73	0.00	27.78	16.67
event-log	0	5	0	8	0	0.00	1.95	0.00	14.81	0.00
file	0	47	3	32	6	0.00	18.36	17.65	59.26	25.00
group	0	19	0	0	0	0.00	7.42	0.00	0.00	0.00
integer-value	0	2	0	3	0	0.00	0.78	0.00	5.56	0.00
loop	0	19	0	16	4	0.00	7.42	0.00	29.63	16.67
multi-state-input	0	62	2	22	2	0.00	24.22	11.76	40.74	8.33
multi-state-output	0	50	2	18	3	0.00	19.53	11.76	33.33	12.50
multi-state-value	3	119	6	45	4	21.43	46.48	35.29	83.33	16.67
notification-class	1	40	0	54	4	7.14	15.63	0.00	100.00	16.67
octetstring-value	0	3	0	0	0	0.00	1.17	0.00	0.00	0.00
positive-integer-value	0	9	0	5	0	0.00	3.52	0.00	9.26	0.00
program	0	29	0	19	2	0.00	11.33	0.00	35.19	8.33
schedule	0	23	1	54	4	0.00	8.98	5.88	100.00	16.67
structured-view	0	6	0	2	0	0.00	2.34	0.00	3.70	0.00
trend-log	0	16	0	29	4	0.00	6.25	0.00	53.70	16.67
trend-log-multiple	0	1	0	2	1	0.00	0.39	0.00	3.70	4.17

Table 4.18: BIBBs and occurrences extracted from selected product listings

BIBBS	Sensor	Controller	Actuator	Advanced Controller	Work station	Sensor Percent	Controller Percent	Actuator Percent	Advance Controller Percent	Work station Percent
AE-ACK-A	0	0	0	3	24	0.00	0.00	0.00	5.56	100.00
AE-ACK-B	1	30	0	54	4	7.14	11.72	0.00	100.00	16.67
AE-AS-A	0	0	0	0	24	0.00	0.00	0.00	0.00	100.00
AE-AVM-A	0	0	0	0	1	0.00	0.00	0.00	0.00	4.17
AE-AVN-A	0	0	0	0	11	0.00	0.00	0.00	0.00	45.83
AE-EL-I	0	5	0	8	0	0.00	1.95	0.00	14.81	0.00
AE-ELV-A	0	0	0	0	1	0.00	0.00	0.00	0.00	4.17
AE-N-A	0	0	0	1	24	0.00	0.00	0.00	1.85	100.00
AE-N-E	0	0	0	4	2	0.00	0.00	0.00	7.41	8.33
AE-N-I	1	37	0	54	4	7.14	14.45	0.00	100.00	16.67
AE-VM-A	0	0	0	0	24	0.00	0.00	0.00	0.00	100.00
AE-VN-A	0	0	0	0	24	0.00	0.00	0.00	0.00	100.00
DM-ADM-A	0	0	0	0	19	0.00	0.00	0.00	0.00	79.17
DM-ANM-A	0	0	0	0	17	0.00	0.00	0.00	0.00	70.83
DM-ATS-A	0	0	0	4	11	0.00	0.00	0.00	7.41	45.83
DM-BR-A	0	0	0	0	6	0.00	0.00	0.00	0.00	25.00
DM-BR-B	0	17	0	22	4	0.00	6.64	0.00	40.74	16.67
DM-DCC-A	0	0	0	0	7	0.00	0.00	0.00	0.00	29.17
DM-DCC-B	4	256	5	54	13	28.57	100.00	29.41	100.00	54.17
DM-DDB-A	1	62	0	52	24	7.14	24.22	0.00	96.30	100.00
DM-DDB-B	12	256	17	54	24	85.71	100.00	100.00	100.00	100.00
DM-DOB-A	0	4	0	14	9	0.00	1.56	0.00	25.93	37.50
DM-DOB-B	12	256	16	53	24	85.71	100.00	94.12	98.15	100.00
DM-LM-A	0	0	0	0	12	0.00	0.00	0.00	0.00	50.00
DM-LM-B	0	29	3	24	13	0.00	11.33	17.65	44.44	54.17
DM-MTS-A	0	0	0	1	24	0.00	0.00	0.00	1.85	100.00
DM-OCD-A	0	0	0	0	4	0.00	0.00	0.00	0.00	16.67

continuation of Table 4.18

BIBBS	Sensor	Controller	Actuator	Advanced Controller	Work station	Sensor Percent	Controller Percent	Actuator Percent	Advance Controller Percent	Work station Percent
DM-OCD-B	0	6	0	10	5	0.00	2.34	0.00	18.52	20.83
DM-R-B	0	20	3	14	3	0.00	7.81	17.65	25.93	12.50
DM-RD-A	0	0	0	0	9	0.00	0.00	0.00	0.00	37.50
DM-RD-B	5	137	5	52	11	35.71	53.52	29.41	96.30	45.83
DM-TS-A	0	2	1	3	18	0.00	0.78	5.88	5.56	75.00
DM-TS-B	3	121	7	54	11	21.43	47.27	41.18	100.00	45.83
DM-UTC-A	0	0	0	3	17	0.00	0.00	0.00	5.56	70.83
DM-UTC-B	0	61	6	45	12	0.00	23.83	35.29	83.33	50.00
DS-AM-A	0	0	0	0	3	0.00	0.00	0.00	0.00	12.50
DS-AV-A	0	0	0	0	4	0.00	0.00	0.00	0.00	16.67
DS-COV-A	0	18	0	15	17	0.00	7.03	0.00	27.78	70.83
DS-COV-B	3	110	11	43	4	21.43	42.97	64.71	79.63	16.67
DS-M-A	0	0	0	0	24	0.00	0.00	0.00	0.00	100.00
DS-RMP-B	0	1	0	0	0	0.00	0.39	0.00	0.00	0.00
DS-RP-A	0	36	0	28	24	0.00	14.06	0.00	51.85	100.00
DS-RP-B	14	256	17	54	24	100.00	100.00	100.00	100.00	100.00
DS-RPM-A	0	8	0	14	24	0.00	3.13	0.00	25.93	100.00
DS-RPM-B	11	211	14	54	15	78.57	82.42	82.35	100.00	62.50
DS-TS-B	0	1	0	0	0	0.00	0.39	0.00	0.00	0.00
DS-UTC-B	0	1	0	0	0	0.00	0.39	0.00	0.00	0.00
DS-V-A	0	0	0	0	24	0.00	0.00	0.00	0.00	100.00
DS-WP-A	0	35	0	30	24	0.00	13.67	0.00	55.56	100.00
DS-WP-B	14	256	17	54	16	100.00	100.00	100.00	100.00	66.67
DS-WPM-A	0	8	0	8	24	0.00	3.13	0.00	14.81	100.00
DS-WPM-B	1	126	5	54	13	7.14	49.22	29.41	100.00	54.17
NM-RC-B	0	2	0	0	0	0.00	0.78	0.00	0.00	0.00

continuation of Table 4.18

BIBBS	Sensor	Controller	Actuator	Advanced Controller	Work station	Sensor Percent	Controller Percent	Actuator Percent	Advance Controller Percent	Work station Percent
SCHED-AVM-A	0	0	0	0	1	0.00	0.00	0.00	0.00	4.17
SCHED-E-B	0	0	0	16	4	0.00	0.00	0.00	29.63	16.67
SCHED-I-B	0	10	0	50	4	0.00	3.91	0.00	92.59	16.67
SCHED-R-B	0	4	0	0	0	0.00	1.56	0.00	0.00	0.00
SCHED-VM-A	0	0	0	0	24	0.00	0.00	0.00	0.00	100.00
SCHED-WS-A	0	0	0	0	8	0.00	0.00	0.00	0.00	33.33
SCHED-WS-I	0	6	1	0	0	0.00	2.34	5.88	0.00	0.00
T-A-A	0	0	0	0	2	0.00	0.00	0.00	0.00	8.33
T-ATR-A	0	0	0	0	13	0.00	0.00	0.00	0.00	54.17
T-ATR-B	0	10	0	26	4	0.00	3.91	0.00	48.15	16.67
T-AVM-A	0	0	0	0	3	0.00	0.00	0.00	0.00	12.50
T-V-A	0	0	0	0	23	0.00	0.00	0.00	0.00	95.83
T-VMT-A	0	0	0	0	6	0.00	0.00	0.00	0.00	25.00
T-VMT-E	0	0	0	3	4	0.00	0.00	0.00	5.56	16.67
T-VMT-I	0	16	0	29	4	0.00	6.25	0.00	53.70	16.67

4.6.8 Network level simulation

Historically, the network level of a BAS requires gateway devices to translate between protocols to allow networking access between devices. Typically, a network controller connects multiple application controllers together, which in turn are connected to a range of underlying sensors and actuators. Protocol convertors and broadcast forwarders sit in between controllers and field devices, or are embedded into the controllers. With a full BACnet/IP network, the topology can follow a more traditional IP-based network. The network topology is considerably flat, with related sensors, actuators and controllers all located within one subnetwork. Each air handling unit can be split into separate subnetworks with BACnet-routers used for cross-subnet broadcasts, or, depending on the size, the full system can be in one subnet. For this simulation, the entire building's BAS is in a single subnetwork. For network capture, all traffic is port forwarded to a capture port, where *Dumpcap* (Combs, 2017) is used to capture the traffic. An external network connection exists to a physical switch, whereby a deployment management system, and the external attacker device are located. Further attacker information is defined in Section 4.6.9.

The BACnet devices and network traffic are defined using the *BACnet open stack* (Karg, 2017), where the profiles described in section 4.6.7 were used to construct the devices which are deployed into lightweight Ubuntu virtual machines. Using the *BACnet open stack* (Karg, 2017) allowed the simulation to be defined in software, rather than being restricted based on the specific vendor hardware controllers in use, providing a more generic simulation.

While the *BACnet open stack* (Karg, 2017) is the most complete of any identified open source BACnet stack, it is described as a baseline system, with the requirement of customising the *BACnet open stack* to implement into a device. A number of core functions were required to be implemented into the *BACnet open stack* to represent the behaviour of the devices for this scenario. These included a method of handling *CoV* requests, identified in Johnstone et al. (2015) as being incomplete, and ensuring each device was represented appropriately. The *BACnet open stack* was amended in the following ways to be used in this scenario.

1. The *CoV* handler function for analog input values was defined in the analog input object definition, by default, only the CoV handler for binary input values was defined.

2. The present value for analog outputs had a range restriction from 0.0 to 100.0, this was removed to allow fan speeds in RPM defined in the scenario to be held in analog output objects.

3. To allow for controller devices to subscribe to other devices, the confirmed *CoV* subscription code was implemented into the device code.

The behaviour of the devices are defined based on best practices ('Best Practices in HVAC Controls', 2012) and analysis revealed by the real data analysis undertaken in §5.2.1. Each air handling unit controller manages a range of sensors and actuators. Namely, each air handling unit polls each duct sensor for its analog input. The analog outputs are used to control the actuators and fans. The values of the analog outputs are calculated based on the input values from the sensors using the control logic defined in §4.6.6. Each thermostat controls an associated VAV box in its zone, where it reads the values of the pressure sensor and flow sensor, and uses the analog output control to position the damper actuator in the VAV box. The communication paths are further detailed in Figure 4.18.

Drawn from the analysis of the real captured data from the university network, the choice of controller impacts heavily on the BACnet commands traversing the network, particularly *Write property* commands. Vendors can implement proprietary middleware network protocols, which encapsulate BACnet commands. These commands are often classed as *commandable properties*, such as write commands. As the real network data captured relies on a closed source middleware BACnet layer for *Write property* commands, it is unclear what real network behaviour would entail for write commands. Thus the assumption is made that write commands to output values, whose values command physical changes in systems do occur; while write commands to input values do not occur given they are used to hold sensor reading values. This assumption aligns with the default behaviour of the *BACnet open stack* (Karg, 2017), and was implemented as the normal behaviour for the simulation scenario. Given the discrepancies between the simulation data, which contains write commands in native BACnet, and the real dataset, which uses a proprietary protocol to encapsulate the majority of BACnet write commands, the simulation data which involves write commands has been omitted for statistical comparisons between the datasets.

Internally, for the simulation to operate the controllers need to make decisions based on the data it holds on each sensor it manages. The logic of each device is defined in §4.6.6. The data read from each sensor is stored in a corresponding input object, which informs the value to be written to actuators from the output objects in the controller. For the simulation, this internal process is undertaken through reading datapoints into the object after receiving the current value from the sensor. This is dependent on timing, as the data is read in every second based on the values generated, while the polling occurs at intervals. The polling interval is different for each vendor and implementation. For example, Shepard (2013) details a one second minimum polling interval, while Thn (2017) defaults its polling interval at two seconds. The real traffic analysis identified a polling interval of eight seconds on setpoint values. Polling is synchronous, the device polls one object then the next in a list. If the network bandwidth is limited, or there are a large number of polls, the network would constantly be polling for values. For this simulation, the values being polled are not static setpoints, rather, sensor values. Additionally, there are three controllers polling seven sensors and two thermostats each, six thermostats polling six VAV boxes, and the logger device polling every device in the network every 15 minutes. As such, an interval of 10 seconds between polls was selected, to reduce the network bandwidth consumed if the minimum polling time was set, and to ensure polling is not a continuous action in the network. Revealed through analysis of the real network traffic detailed in §5.2.1, each object subscription has a subscription duration of five hours, with re-subscriptions occurring every 20 minutes. This behaviour was replicated in the simulation scenario.

Some BACnet systems implement a logging device, which retrieves values from other devices in the network for long term storage, either in a database or flat file. This type of logging often relies on the *trendlog* object being implemented in a device. As noted from the analysis of Internet-facing devices in §4.2, only supervisory/control devices typically implement *trendlog* objects. Alternatively, a logging device could use normal polling via the *Read property* or *Read property multiple* service to retrieve values from sensors and actuators, in addition to controllers. The polling approach was undertaken for the simulation scenario, whereby a logging device polled the present value property in each object in each device every 15 minutes.

The simulation was run on a VMware ESXI 6.0 Server, which had 32GB of RAM and an AMD 3GHz 8 core processor, totalling 24GHz processing power for the virtual machines housed on the server. The normal simulation consisted of 59 virtual machines running for 31 days. There were seven classes of virtual machine, six for the types of devices running in the simulator, and one type for capturing the network traffic. The simulation consisted of 21 actuator devices, 21 sensor devices, 3 controllers, 6 thermostats, 6 VAV boxes, 1 workstation and 1 logging device. The specifications for each device type are detailed in §3.7, Table 3.10. The average load of the ESXi Host CPU per virtual machine was 200 MHz, the average load of the ESXi Host memory per virtual machine was 150 MB for sensors, and 400MB for controllers and thermostats. With a baseline defined and used to generate network traffic classed as normal, an attack framework was developed to generate malicious traffic for this scenario.

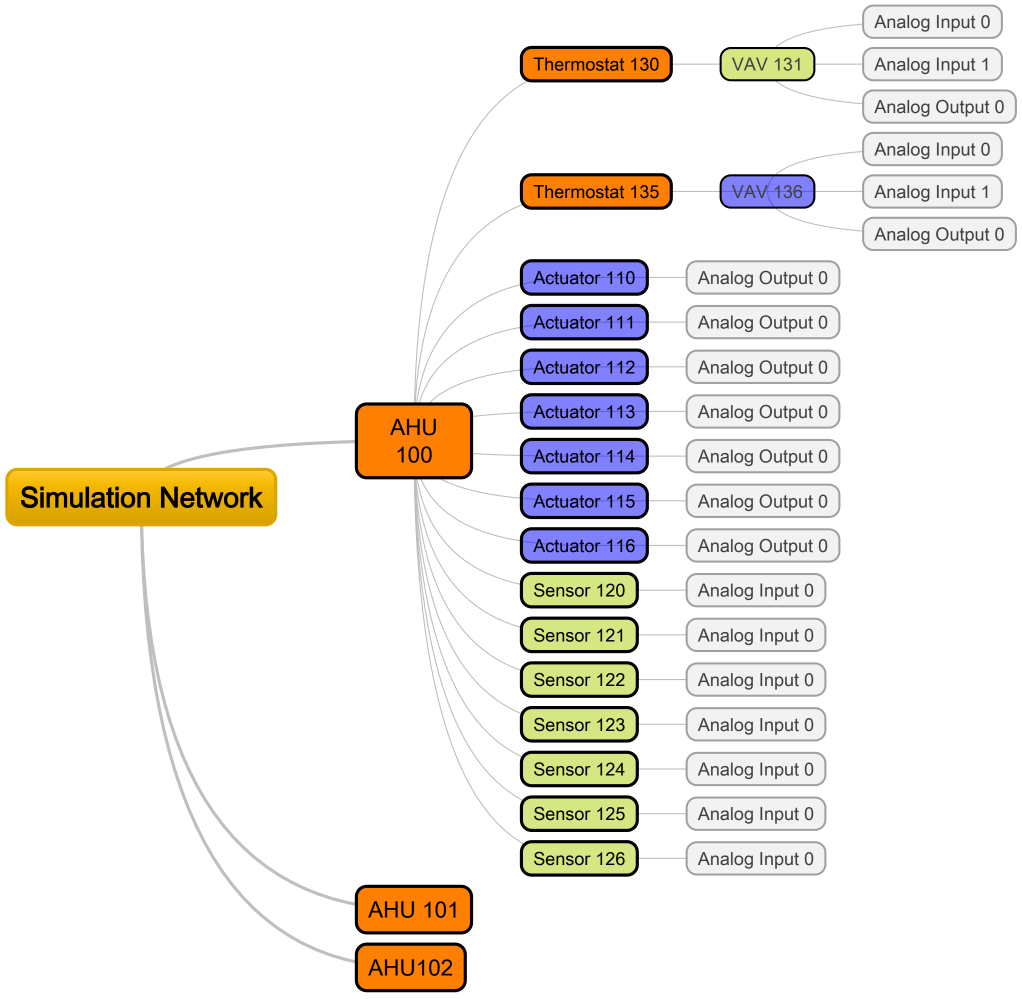


Figure 4.18: Logical mapping of simulated devices and the prominent underlying BACnet data structures

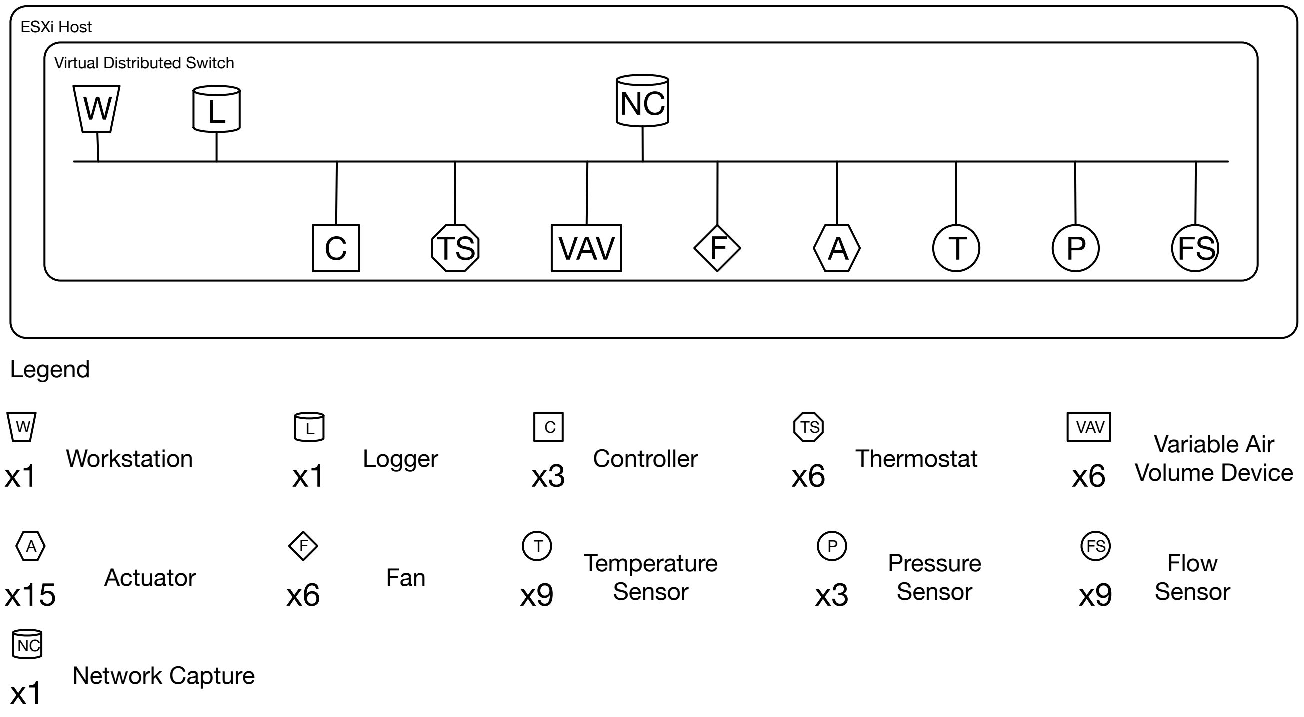


Figure 4.19: Network topology of the defined simulation scenario

4.6.9 Attack implementation

To generate malicious traffic for the purpose of testing intrusion detection methods, a range of attacks were required to be implemented. Analysis of the attacks identified in §4.3, reveals which attacks have previously been used in the literature to identify malicious activity. Of the 26 identified known attacks, 8 have previously been used for testing anomaly detection methods. Many of the identified attacks utilise application and network layer messages regarding BACnet routers. The scenario testbed implemented does not implement BACnet routers, additionally, from the real data collection, it was identified that router related commands are very rare, outlined further in Table 5.3. Therefore, the 10 router based attacks are excluded from testing. Additionally, attacks using malformed packets were discounted as the aim of the research is to identify normal packets causing malicious actions. Further, the services *create object* and *delete object* are not implemented services in the open *BACnet open stack* (Karg, 2017), used for the simulation testbed. Expansion of these features was deemed out of scope for the research given the existence of more frequently used services which could be deemed malicious, and thus two additional attacks were not investigated. Finally, source manipulation for the *Smurf attack* identified was deemed very similar to a spoofing attack using an *I-am* command, and thus exists as part of the *I-am* spoof attack. The remaining identified attacks from §4.3, in addition to the priority array overwrite identified in §4.3.1 were used to form the attack framework.

Some commands can have multiple potential behaviours. For example, a reconnaissance attack using the *Who-is* command could request all devices at once, or each device individually with pauses between requests. Therefore, behaviours for each attack were classified based on frequency as outlined in Table 4.19.

As defined by the device profiles, only certain devices can initiate services by default. Specifically, the majority of sensor and actuator devices analysed do not permit initiating commands which cause action in other devices. The limited functionality of sensor devices reduces the risk they pose to other devices in the network, however, it cannot be overstated that the device profiles outline the minimum functionality defined for classification as the device type. Nothing prevents vendors from implementing additional services into their devices (Distech, 2010). However, noted through the analysis of Internet-accessible devices in §4.6.7.1, the number of sensors and actuators who can initiate commands is low.

For the scenario in this research, there are three malicious devices in the network which all have different capabilities due to their device classification.

1. One external device, which has not previously interacted with the system, and can undertake reconnaissance and Denial of Service attacks.
2. One internal sensor, which has limited capabilities, but can undertake some reconnaissance and flooding attacks.
3. One internal controller, which has initiate capabilities and thus can perform all classes of attack.

The internal devices continue to operate as normal, with sporadic malicious normal commands occurring. To counter potential self bias, attacks are pseudo-randomly selected by the adversary device prior to commencement. A random value is drawn form the *Python* random module and set as the random seed for further random module calls for each attack instance. Each instance of each command has a random seed generated for drawing from a range of random sleep times, making the time between each malicious yet normal packet have no immediately similar pattern. Further, the order in which properties are read by the attack code is also shuffled, to remove bias from implementation. All seed values are recorded for future repeatability. Generating the attack traffic dataset consisted of repeating the first week of the defined scenario with the addition of the randomised malicious actions occurring.

Table 4.19: Attack framework for simulation scenario, outlining attacks, adversary launch point and network target

Type	Class	Command/Property	Detail	Adversary Device	Target Device	Anomaly Class
Frequency	Flood	Who-is	-	Controller, Sensor	Network	Known-Unknown
Frequency	Flood	I-am	-	Sensor	Network	Known-Unknown
Frequency	Denial of Service	I-am	Smurf Attack	Sensor	Controller	Known-Known
Frequency	Reconnaissance	Who-is	Un-Restricted Who-is	External, Sensor	All	Known-Unknown
Frequency	Reconnaissance	Who-has	-	External	All	Known-Known
Frequency	Reconnaissance	Read property multiple	-	Controller	All	Known-Known
Frequency	Write	Write property	In Bounds	Controller	Actuator	Known-Unknown
Frequency	Write	Write property	Out of Bounds	Controller	Actuator	Known-Known
Non-Frequency	Denial of Service	Subscribed Change of Value	CoV Disconnect	Controller	Thermostat	Unknown-Unknown
Non-Frequency	Denial of Service	Reinitialize device	-	External	Controller, Actuator	Known-Known
Non-Frequency	Reconnaissance	Who-is	Individual Devices	Controller, Sensor	All	Unknown-Unknown
Non-Frequency	Reconnaissance	Read property	Individual Properties	Controller	All	Unknown-Unknown
Non-Frequency	Spoofing	I-am	-	Sensor	Controller	Known-Unknown
Non-Frequency	Write	Write property	Priority Array	Controller	Actuator	Unknown-Unknown

4.6.9.1 Flooding attack

The flooding attack class consists of repetitively sending a command across the network with the aim of slowing traffic and/or preventing devices from operating correctly. In BACnet, any service could be used for this purpose, however, for this scenario the *I-am* and *Who-is* services are selected. The *I-am* service is used to answer unconfirmed broadcast messages from other BACnet devices when queried for a device identifier from the *Who-is* service. As there is no authentication for the *I-am* function, any device can claim to be any number of other device identifiers. Similarly, the *Who-is* attack sends a wide range of device identifiers over the network, where legitimate devices can then respond causing more traffic. The flooding attacks loop through a range of device identifiers, requesting for each identifier, with the aim to disrupt the network. For the purpose of this simulation, these attacks occur pseudo-randomly for a variable range of time.

4.6.9.2 Spoofing attack

For a more targeted attack, the *I-am* command is used to trick specific BACnet devices into sending unintended traffic towards it. For this attack scenario, a specific device is selected to spoof, when activated, the device claims to be a different device to cause traffic to be sent to itself. A potential issue with the spoofing attacks is preventing the real device from sending back an *I-am* and the interaction of two devices claiming a single device ID. It was expected that this behaviour will be vendor device specific, as the BACnet standard claims no handling of misconfigured devices (SSPC-135, 2012). For the *BACnet open stack* (Karg, 2017), if a device identifier was saved in the address cache of the device, claiming to be a device did not change which device was sent the message. If the device had no address cache, the first device to claim a device identifier was sent the message.

4.6.9.3 Write attack

Three write attack class scenarios were defined for the framework. First, with in-bounds value sent repeatedly to a specific device, similar to the attack undertaken in §4.4.1. Second, an out-of-bounds write attack, where the values are not what should be expected by the device. To determine the behaviour of the device defined in the simulation, a range of values were selected for writing to each object in various data formats. One each of a control character, negative value, string value and long integer were sent to the present value property of each object type operating in the simulation. The *BACnet open stack* (Karg, 2017) provides error handling and range limits for some objects. In total, 224 out-of-bounds command combinations were constructed, using the four data inputs, seven object types and eight data type tags. Each command was sent to the default server device in the *BACnet open stack* for testing (Karg, 2017). 24 out of bounds values were accepted by the stack with varying changes to the input stored in the object, see Table 4.20. 22 of the commands resulted in a null or zero value writing into the present value property of the object, regardless of the input. One command written to the *Analog value* object was written directly, while another was incorrectly written due to a float precision error. Finally, the priority array write attack identified in §4.3.1 and discussed in §4.3.2 was implemented.

Table 4.20: Results from out-of-bounds values sent to specific objects

Object Type	Data Type Tag	Value to Write	Value sent
Analog Output	0 (Null)	\x03	NULL
Analog Output	4 (Real)	\x03	0
Analog Output	0 (Null)	-1	NULL
Analog Output	0 (Null)	a	NULL
Analog Output	4 (Real)	a	0
Analog Output	0 (Null)	9999999999999999-9999999999999999	NULL
Analog Value	4 (Real)	\x03	0
Analog Value	4 (Real)	-1	-1
Analog Value	4 (Real)	a	0
Analog Value	4 (Real)	9999999999999999-9999999999999999	10000000331813535-1409612647563264
Binary Input	9 (Enumerated)	\x03	0
Binary Input	9 (Enumerated)	a	0
Binary Output	0 (Null)	\x03	NULL
Binary Output	9 (Enumerated)	\x03	0
Binary Output	0 (Null)	-1	NULL
Binary Output	0 (Null)	a	NULL
Binary Output	9 (Enumerated)	a	0
Binary Output	0 (Null)	9999999999999999-9999999999999999	NULL
Binary Value	0 (Null)	\x03	NULL
Binary Value	9 (Enumerated)	\x03	0
Binary Value	0 (Null)	-1	NULL
Binary Value	0 (Null)	a	NULL
Binary Value	9 (Enumerated)	a	0
Binary Value	0 (Null)	9999999999999999-9999999999999999	NULL

4.6.9.4 Reconnaissance

An adversaries goal of reconnaissance can be achieved through a range of commonly operating services. To determine which devices are operating on the network, five different scenarios were constructed. The *Who-is* command was used without a limiting device range, with the aim of requesting every device in the network to respond. Similarly, the *Who-has* command was used to request a range of object identifiers, whereby devices which hold the specific object identifier would respond. The *Who-has* is the object equivalent of the *Who-is* command, where a broadcast is sent to request either object names or object IDs. Any device which has a request object name or object ID will broadcast a response. For the purpose of this simulation, the range of object identifiers to enumerate is restricted, as undertaking a full linear search of all device identifiers and object identifiers in the simulation network would result in 13,363,049,358 requests. The range is restricted to eight object types, and six object identifier numbers, resulting in 2,736 requests.

The last frequency based reconnaissance command used was the *Read property multiple* command, which reads all properties from a device in the network in one transaction.

The network typically requests specific device addresses using *Who-is*, as such a non-frequency based command behaviour was defined which requested a range of existing device identifiers in a random order, with different timing between each request. The same approach was taken with the *Read property* command, where only existing properties are read from a device.

4.6.9.5 Denial of Service

A device can be denied service through sending a *ReinitializeDevice* service, which essentially restarts a device. *ReinitializeDevice* is typically undertaken by a human operator through a workstation machine, and it is recommended to be a password protected action. For this scenario, it is an assumption that this password has been obtained by the adversary for the *ReinitializeDevice* to occur. Given the lack of default encryption on BACnetworks, any device which is listening on the network could easily capture the password when a legitimate *ReinitializeDevice* command is used, as the password would be travelling unencrypted over the network. Given the sensitivity of this service, alerts should be generated whenever this command is executed, making it far easier to find than the other services, as it is a service that is only used in specific situations executed by a human operator, and is thus uncommon. The *BACnet open stack* (Karg, 2017) implementation of *ReinitializeDevice* does not restart the device, but provides the correct network transaction over the network. This allows for the network action to be demonstrated, but not impact the device, providing a means to capture this class of network attack without a long-term effect on the simulation environment. The last class of attack tested was the *CoV* subscription theoretical attack, initially tested in Johnstone et al. (2015), and further modelled in Peacock et al. (2018). The extended scenario discussed in Peacock et al. (2018) was undertaken, whereby multiple devices are subscribed to the attacker devices target, to determine if a delay in *CoV* notifications to other hosts occurs.

4.7 Chapter summary

This chapter detailed phases one, two, and three of the research design as described in §3.5, with the aim of understanding and identifying issues in the BACnet/IP protocol. First, the BACnet standard was reviewed, identifying commands which could be used for malicious means. Next, device details from three years of network scans were retrieved from Censys (Censys, 2018), and analysed for device types, and profiles. Following, known attacks from the literature were retrieved, coupled with the identified potential attacks, and formed into a threat model using the STRIDE threat matrix. The threat model was then applied to a controller device, to identify which commands used in each attack posed the largest threat to controller devices. The identified commands were then compared to the device profiles extracted from the network scans to summarise the potential risk posed to open BACnet devices. The section concludes with modelling and implementation testing of the Priority Array overwrite attack.

Next, a range of BACnet/IP simulators were identified and tested for configurability to implement a range of scenarios for data generation. To test each simulator, two datasets (trial and theory-test) were generated using basic scenarios and some malicious traffic. The Trial Dataset was used to evaluate if a machine learning algorithm could detect BACnet specific network attacks. A back-propagation Artificial Neural Network was tested against a frequency based *Write property* attack, with 100% detection rate. Further, a flow analysis method using command frequencies was tested, with positive visual identification of the attack occurrences. The Theory Test Dataset was generated to assess further simulators, and implement the theoretical *CoV* attack identified. The results were positive, but required a larger dataset for further testing.

With the requirement for more data identified, data was collected from a real BACnetwork, to understand generally how network controllers communicate. Further, a larger testbed for data generation was designed and implemented, using a defined simulation scenario, algorithms to generate data, and a BACnet/IP network topology. Finally, an Attack Framework was defined from the applicable attacks identified in §4.3. The Attack Framework was then generated and tested, before being implemented into the testbed scenario for malicious traffic generation.

Chapter 5

Results

This chapter describes the application of a range of defined experiments undertaken against the three major datasets, denoted as the Real Normal, Synthetic Normal and Synthetic Attack sets. First, a generic preprocessing method is discussed for the network capture files used. Next, a range of unsupervised methods were used to explore the structure of the Real Normal Dataset, including community detection, a comparison of clustering approaches, and time series analysis. Then, the results of the unsupervised methods applied to the Synthetic Normal Dataset, along with a discussion comparing the two Normal Datasets are presented. Following, the unsupervised methods were applied to the Synthetic Attack Dataset, with comparisons between the Synthetic datasets drawn to highlight the differences between normal and malicious network traffic in the simulation. With the differences stated, the results of building, training and testing a range of Hidden Markov Models defined using ten equally-sized samples drawn from the Synthetic Normal Dataset for training, and a shared testing sample from the Synthetic Attack dataset are elaborated upon. Finally, a comparison of evaluation metrics for the six Hidden Markov Model approaches are presented using six hosts as examples.

5.1 Preprocessing

Preprocessing was undertaken for each dataset to present the data in the necessary format for each experiment. The raw data for each dataset consisted of multiple *pcapng* network capture files, which contained both BACnet and other network protocol traffic. Each *pcapng* file was filtered and split into files containing only BACnet traffic, and finally merged into a single file using *Tshark*. The *Wireshark* native BACnet statistics plugin was then used via *Tshark* to output frequency values at a service level. Further manipulation was undertaken using *sed*, before being loaded into a *Python Pandas dataframe* for application of each method. For the real network dataset, IP address anonymisation was undertaken using *pycryptopan 0.01* a *Python* implementation of the Crypto-PAn algorithm, a prefix-preserving anonymisation tool (Bauer, 2012). For each dataset, flow records were generated using the CERT NetSA Security Suite utility *YAF* (Yet Another Flow meter) (Trammell, 2018), into *IPFIX* format flows. Further, the bounded limits for generating the flows were set to 300 seconds for the Idle-Flow-Timeout and 1800 seconds for the Active-Flow-Timeout.

5.2 Real Normal Dataset

The Real Normal Dataset is comprised of one month of real network data collected from an Australasian university. All hosts in the network are controller-level devices, with no sensor data represented. Additionally, part of the network implementation uses a proprietary middleware implementation of BACnet/IP, which was out of scope of the research conducted.

5.2.1 Real Normal Dataset: Analysis

To visualise the full network, the service counts between hosts were derived from the data collected and used to generate a directed network graph, detailed in Figure 5.1. Hosts were defined as nodes, with network services, and network service counts used between each host as the edges, and edge weights respectively. Community detection has been used in cyber security for identifying critical nodes in a network, to better target defences. As such, a range of community detection algorithms were explored and applied to the network. To identify the importance of hosts in the network, the degree of centrality, and the eigenvector centrality were used. Further, modularity, a measure of network and graph structure which can measure connections between nodes was used. Modularity is the density measurement of edges inside a community compared to edges connected to outside communities, minus the expected fraction if edges were distributed randomly, with values ranging from -1 to +1. A positive value describes that the edges in the group exceeds the number expected based on a random distribution of edges. The Louvain community detection unsupervised algorithm (Blondel, Guillaume, Lambiotte & Lefebvre, 2008) was used against the network graph, revealing three communities, with a modularity ranking of 0.238 using the standard modularity resolution of 1.0. The eigenvector centrality was calculated with 100 iterations of the *Gephi* graph algorithm package (Gephi, 2017). Larger nodes in the graph represent higher eigenvector centrality values. The degree (total connections) and eigenvector centrality values for the network are detailed in Table 5.1.

Figure 5.1 shows the observed connection between all hosts, the degree of interaction between hosts, and the centrality of specific hosts. Specifically, it can be seen that host 246.35.6.57 sends a significant amount of network traffic to the subnet broadcast address (anonymised to .254). Unsurprisingly, the controllers of interest in the monitored building have the higher eigenvector values. However, what was not anticipated was the regularity of individual controllers communicating between controllers located in other buildings. Through evaluation of the network graph, the large amount of outbound traffic from host 246.35.6.57 with no inbound connections was a prime candidate for a misconfiguration, or potential network attack, and was investigated further. The community detection algorithms applied identified a range of underlying, connected hosts, which align with the understanding of the network.

Table 5.1: Hosts exhibiting measures of centrality, modularity and clustering coefficients for the Real Normal Dataset

IP Address	Eigenvector centrality	Indegree	Outdegree	Degree	Modularity Class	Clustering Coefficient
246.35.6.165	1	27	23	50	2	0.29064
246.35.6.172	0.885535	21	18	39	2	0.359684
246.35.6.174	0.884594	13	15	28	2	0.52381
246.35.6.254	0.598651	65	0	65	1	0.017788
246.35.6.80	0.320134	8	10	18	2	0.266667
246.35.6.24	0.159789	2	7	9	1	0.238095
246.35.6.214	0.150417	2	3	5	1	0.666667
246.35.6.49	0.150417	2	4	6	2	0.666667
246.35.6.56	0.150417	2	7	9	2	0.285714
246.66.248.219	0.150417	2	4	6	2	0.666667
246.35.6.52	0.150417	2	6	8	2	0.266667
246.34.95.8	0.150417	2	7	9	2	0.190476
56.0.15.254	0.058332	8	0	8	0	0
246.35.6.71	0.052616	2	3	5	2	0.333333
246.35.6.160	0.052616	2	4	6	2	0.333333
246.35.6.15	0.03973	3	1	4	1	0.25
246.35.6.63	0.025817	3	2	5	2	0
246.35.6.57	0.014989	4	2	6	1	0
246.35.6.62	0.001163	1	3	4	1	0
246.35.6.25	0.001163	1	1	2	1	0
246.35.6.60	0.000832	1	2	3	1	0.333333
246.35.6.30	0	0	3	3	1	0.333333

Further analysis was undertaken through flow analysis of the network communication. A flow is a series of frames describing a conversation between two hosts. Flow analysis ignores the content of messages, and rather infers knowledge from the meta-properties of the conversation, such as start time, end time, frame size and packet occurrences. Further information can be generated from these properties, including session duration, and average packets per second. A total of 39,478,148 frames were processed into 56,613 flows for the 51 communicating hosts, outlined in Table 5.2. 56% of these flows (32,241) were uni-directional flows, which highlights the large amount of broadcast traffic in BACnetworks. 31,560 flows reached the Active-Flow-Timeout value, and 25,028 flows ended due to the Idle-Flow-Timeout being reached. The last 25 flows of the capture period ended prematurely due to the end of the capture file being reached. As denoted in Table 5.2, host 246.35.6.57 has a large proportion of total packets compared to total flows, indicating that each flow has a long duration and many packets.

Analysis of the defined flows was undertaken using the *Python Pandas* library. The aim was to identify common patterns and infer the sequences of commands being sent over the network from these basic features. Each bi-directional flow can contain a sequence of commands occurring in the network split by the flow specification features discussed (active and idle time). These natural sequences defined by time could reveal contextual patterns for the specific network in terms of the normal order of network commands between hosts.

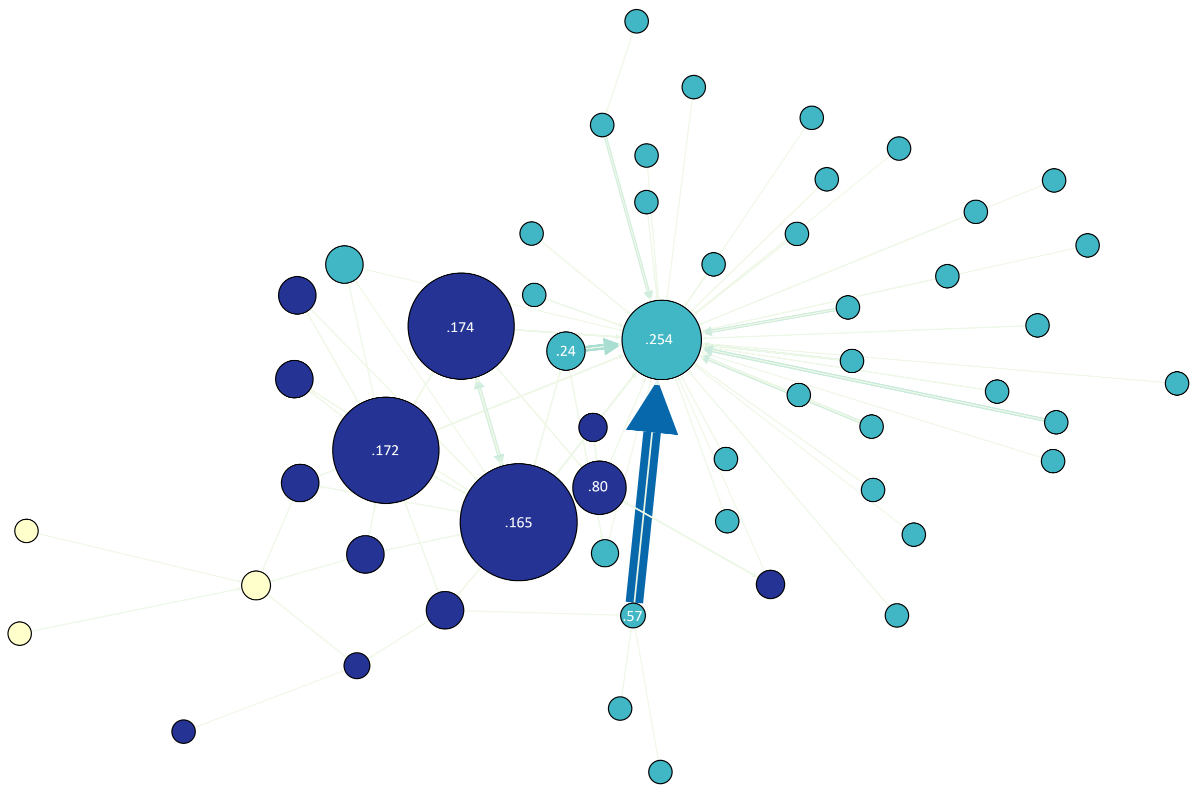


Figure 5.1: Real Normal Dataset directed network graph

To identify patterns in the network flows, unsupervised clustering was undertaken. Two clustering algorithms were used with varying cluster sizes. First, k-means clustering was explored using the two features; number of packets in flow and session duration. K-means requires the cluster size *a priori*, as such, the count of unique commands each host used was selected as a naïve cluster size. Further, a k-means cluster can be influenced by its starting random centroid position. As such, the k-means clusters were calculated using 1,000 different seed values for each input. The occurrences of each cluster counts was determined with the first seed value, which matches the highest occurring cluster value set, selected. For example, for host 246.35.6.80 with four clusters, all 1,000 iterations of the starting seed value returned the same clustering pattern, thus the random seed selected was zero. An example of a k-means cluster for host 246.35.6.172 is detailed in Figure 5.3 top. With a larger number of clusters, varying the starting state increased the variance in cluster labels, an after effect of edge-cases between adjoining clusters. Thus the selection of starting state based on naïve occurrence counts for cluster size was not as certain when larger cluster numbers are selected.

Of interest, host 246.35.6.11 held three flow records, consisting of nine packets which contained two commands. The clustering size selected is thus two, however, upon inspection of the scatter graph there appeared to be only one class. The k-means algorithm could not split this hosts traffic into two clusters as they exhibit the same base properties for the selected features. This highlights the limitations of clustering as a whole, the semantic meaning of the clusters are not necessarily

Table 5.2: Outline of generated flows and packet counts for a range of hosts in the Real Normal Dataset scenario, Host 246.35.6.57 has a high proportion of total packets compared to flows

Host	Flows	Packets	Total Flow %	Total Packet %
246.35.6.80	13329	865020	23.54406%	2.19114%
246.35.6.24	10648	4040832	18.80840%	10.23562%
246.35.6.165	5194	2363202	9.17457%	5.98610%
246.35.6.174	4452	7012981	7.86392%	17.76421%
246.35.6.62	1518	1744548	2.68136%	4.41902%
246.35.6.164	1488	133800	2.62837%	0.33892%
246.35.6.173	1488	133794	2.62837%	0.33891%
246.35.6.175	1488	133800	2.62837%	0.33892%
246.35.6.18	1487	308184	2.62661%	0.78064%
246.35.6.27	1487	231114	2.62661%	0.58542%
246.35.6.16	1487	803085	2.62661%	2.03425%
246.35.6.19	1486	2137281	2.62484%	5.41383%
246.35.6.31	1486	1817463	2.62484%	4.60372%
246.35.6.30	1486	537453	2.62484%	1.36139%
246.35.6.57	**1484**	**13567584**	**2.62131%**	**34.36733%**
246.35.6.60	1482	1875513	2.61777%	4.75076%
246.35.6.172	1181	535355	2.08609%	1.35608%
246.35.6.71	1073	810329	1.89532%	2.05260%
246.35.6.68	857	102081	1.51379%	0.25858%
...	...	...	...	...
Total:	56,613	39,478,148		

correct, based on the distinguishing features selected. Further features would be required, which are not available from a basic network flow.

The second clustering algorithm explored was a Gaussian Mixture Model (GMM). The same features, flow packet size and session duration, were also selected. A GMM can have the cluster size defined *a priori*, or use a maximisation function to derive an appropriate number of clusters. GMM clustering was undertaken using both methods. An example of GMM clustering using the same cluster numbers as the k-means approach is detailed in Figure 5.4 top.

To improve on the naïve cluster values a model was constructed for each host with cluster sizes between 1 and 30. The Akaike information criterion (AIC) and Bayesian information criterion (BIC) were used to compare each model with the aim of identifying local and global optima in cluster size selection. The comparison of the models for a range of hosts is detailed in Figure 5.2. Of note, some hosts had insufficient flow records to fully enumerate a 30 cluster model, and thus were explored with a smaller range of cluster sizes. The k-means and GMM clustering approaches were then re-run using the selected global minima for each model, and detailed in Figure 5.3 and Figure 5.4

The variance of state occurrences in host 246.35.6.71 causes each run to have a unique starting state, and thus each run classifies the data differently. Originally, it was thought that a lack of flows may be a cause of this, however host 246.35.6.24 has a maximum state occurrence of 5, but has an order of magnitude more flow data. With larger cluster sizes, the start state seems to have a more prominent effect on the clustering approach, due to the added variance in classes. Analysis of the GMM step clustering to determine optimal cluster size was informative, but also non-authoritative. The maximisation clustering step-approach potentially overfits on some hosts.

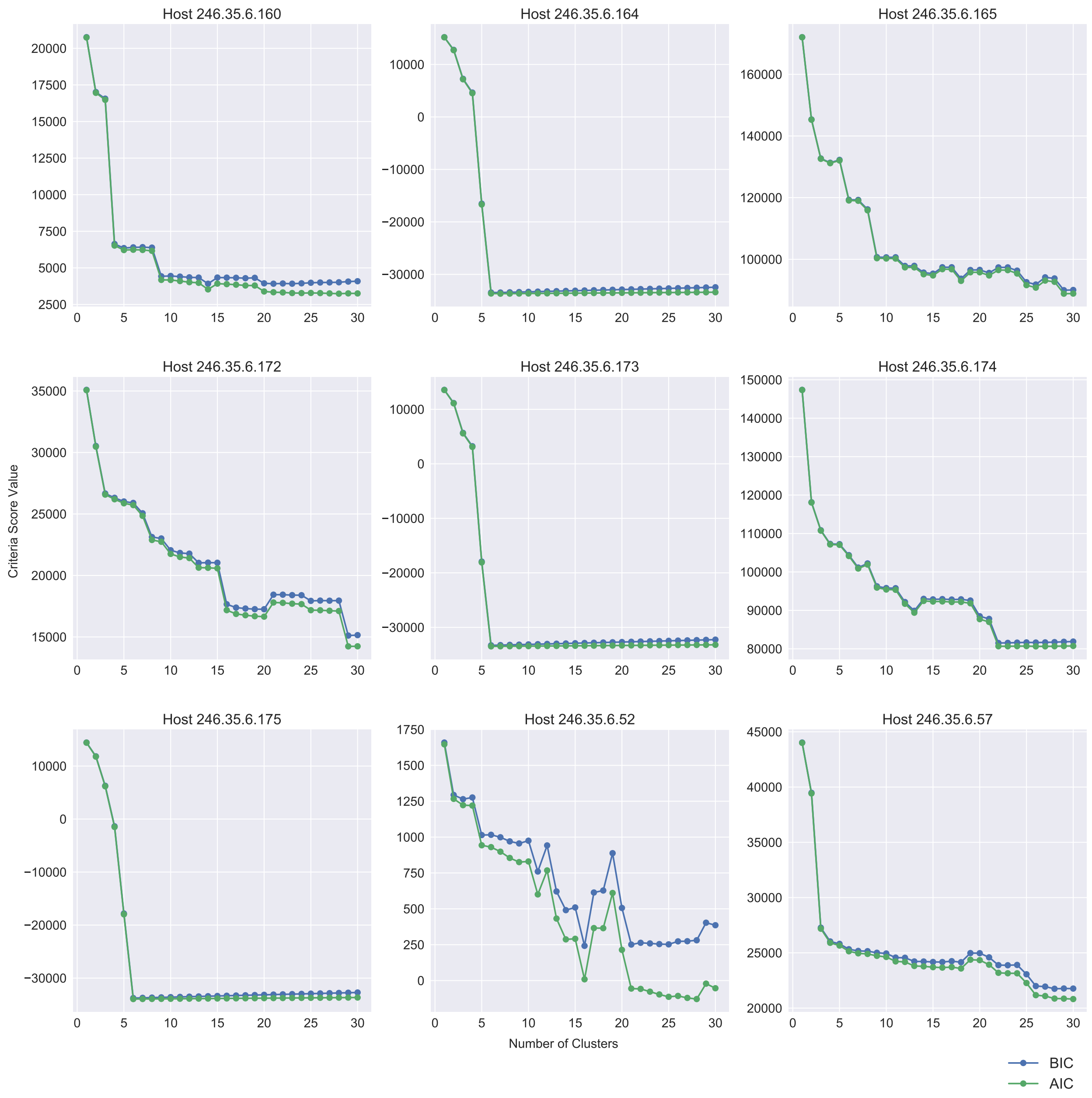


Figure 5.2: Comparison of AIC and BIC cluster size optimisations for selected hosts in the Real Normal Dataset

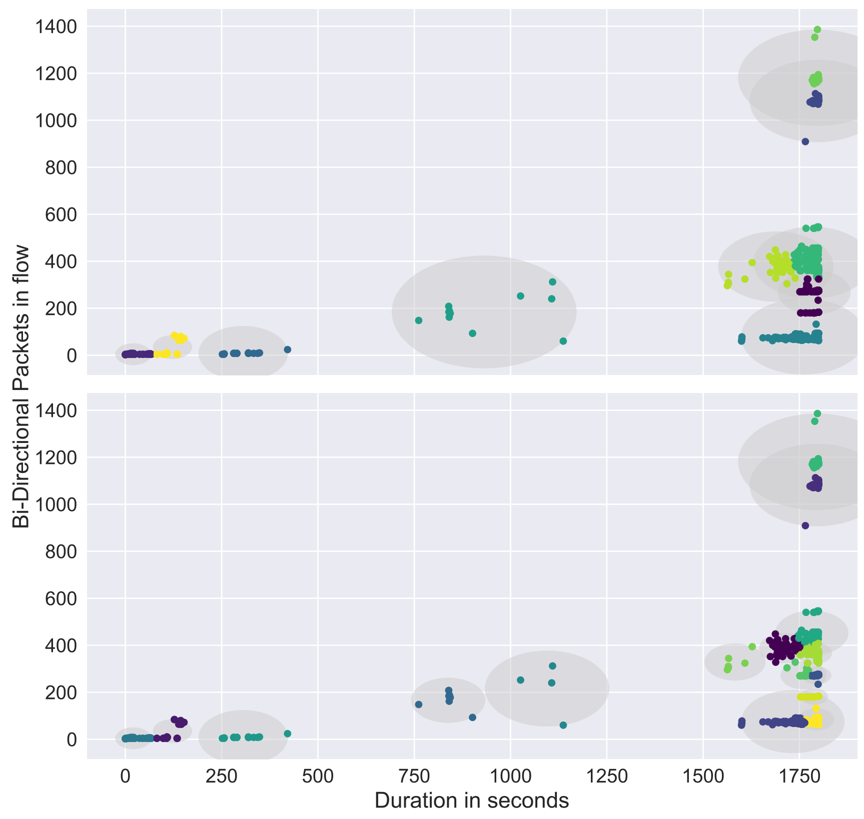


Figure 5.3: Comparison between naïve (k=10) and optimised (k=16) k-means clustering approach for host 246.35.6.172

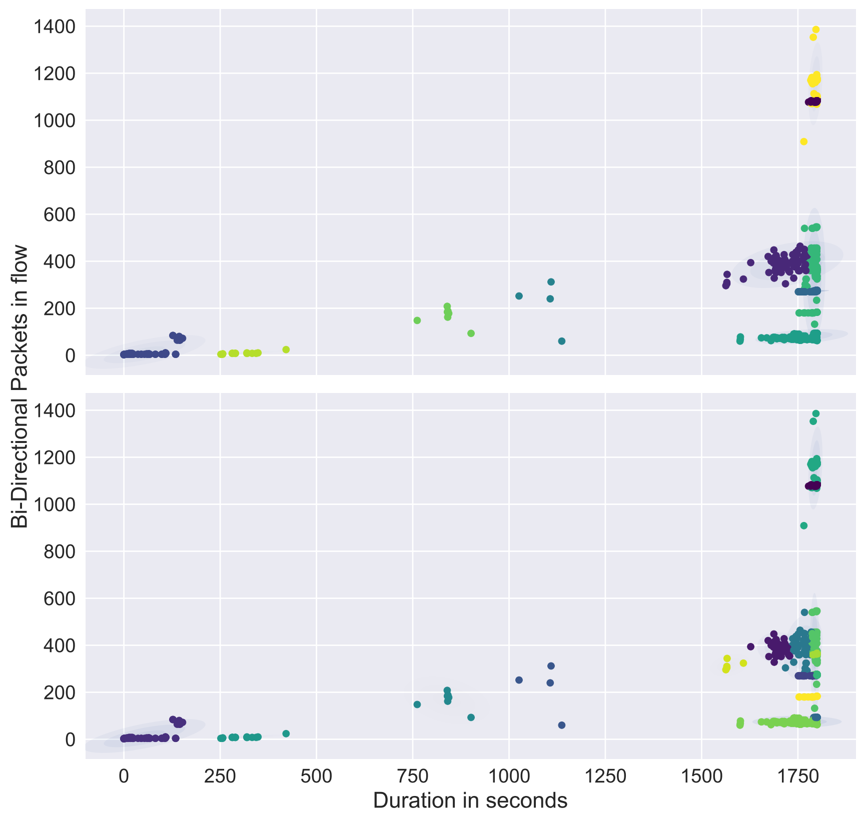


Figure 5.4: Comparison between naïve (k=10) and optimised (k=16) GMM clustering approach for host 246.35.6.172

There is, however, a tradeoff to set between the number of clusters and a minimised error rate criterion value. When the value differential is small, but the number of clusters between two points is large, a selection is required based on interpretation. For example, in host 246.35.6.30, the minimisation of the criterion for the number of clusters analysed is 30, but the difference between six clusters and 30 clusters is 1,000, compared to the difference between five and six clusters being 28,000. In this scenario, a selection of six clusters, opposed to 30 clusters could be justified.

Based on the clustering schema for host 256.35.6.28, when using the selected commands of size three, the next cluster size is a significant reduction, which also falls into a local minima. As with the k-means method for host 246.35.6.11, the minimum clusters from the AIC/BIC is one, however there are two commands from this host. The sample size is comparatively small however (three flows), leaving little distinction between the two types of commands which have the same characteristics in these occurrences. For hosts 246.35.6.78 and 246.35.6.77, the criterion is minimised at the same cluster size as the number of commands issued by the host. Further, the criterion increases after this point, inferring the commands sent by the host are distinguishable enough to be used as the cluster size. For some hosts, the sample size is too small to derive robust cluster sizes. The optimisation processes reduce down to each flow record classed as its own cluster, specifically in host 246.34.95.8, there are seven commands used by the host and 12 flows, the AIC and BIC minimise at 12 clusters. There is clearly information lost, however a number of flows are closely grouped, implying the same type of command sequence.

A major problem with unsupervised methods can be the lack of ground truth. Specifically for the flow data extracted, as each instance is a sequence of events and the sequences are constructed using the same variables across the entire dataset, there is room for misclassification inside the flows. Additionally, the initial cluster sizes are selected based on the number of commands the host undertakes during the duration of the capture. As stated, flow records could contain more than one command type, and thus the combinations of command sequences in each flow could increase the number of unique clusters beyond the base value selected.

The purpose of clustering for this research was to discover natural groups in datasets. Flow clustering has provided an overview of sequences of commands which occur in the network. However, with the various additional factors not captured by the flow data, the clustering approach is not definitive for classifying network command sequences. Further insight was obtained through investigating the sequences of individual packets in the network.

The dimensionality of data increases substantially when progressing from flow level to packet level data. For the Real-Normal dataset, 39,478,148 BACnet packets were transmitted over the network between the 51 hosts. The context provided by this data provides the level of detail for increased analysis, as opposed to the flow data. As the packet level traffic is parseable and each can be assigned a class based on a range of features labelling the packets in a semi-supervised method is possible. Labelling can be undertaken using a range of data transformation methods, examples include data structures such as *JSON*, *XML* or text-based representations which allow the use of text searching methods such as regular expressions. Previous labelling was undertaken in §4.4.1 using *JSON* constructions of each packet for feeding an ANN for classification. The dataset used for the proof of concept was significantly smaller, but resulted in large interim data quantities when processing between formats. The analysis tool chosen for this dataset was the *Python* library *Pandas*, which can use both raw text and *JSON* based data. A text based approach was selected due to a reduced interim data size, with the largest non-reduced host capture text file being 1.4GB, as opposed to the smaller dataset with greater than 1GB *JSON* files. Further, the text data provided the possibility of using regular expressions and *Unix* commands such as *sed* for labelling. The core features of interest for each packet were packet time of arrival, source IP address, destination IP address, packet size, and BACnet contextual information including command, and additional command properties such as sequence numbers, BACnet addresses and object names. These features were extracted using *Tshark*, and parsed using regular expressions and *sed* into comma separated value (*csv*) format files containing bi-directional packets for each host. The major classification feature was the command sent by the source address.

Per host data was preprocessed to anonymise the network addresses. Scatter plots of destination IP and packet size were created for each labeled command sent by the source IP. An example scatter graph is defined in Figure 5.5 The scatter graphs identify very few size variances between packets sent to the same host for each command. The size differences can be accounted for by the address size of the BACnet object identifier being queried in the device, or the difference between sending a *Who-is* broadcast command looking for a specific BACnet ID, or all BACnet devices. The percentage breakdown of commands sent over the network is detailed in Table 5.3. Further investigation at a packet level was undertaken to extract command frequencies. Figure 5.6 outlines the breakdown of commands for host 246.35.6.172. For the outlined host, the *CoV* command traffic is relatively static, while the *Who-is* command causes the change in phase.

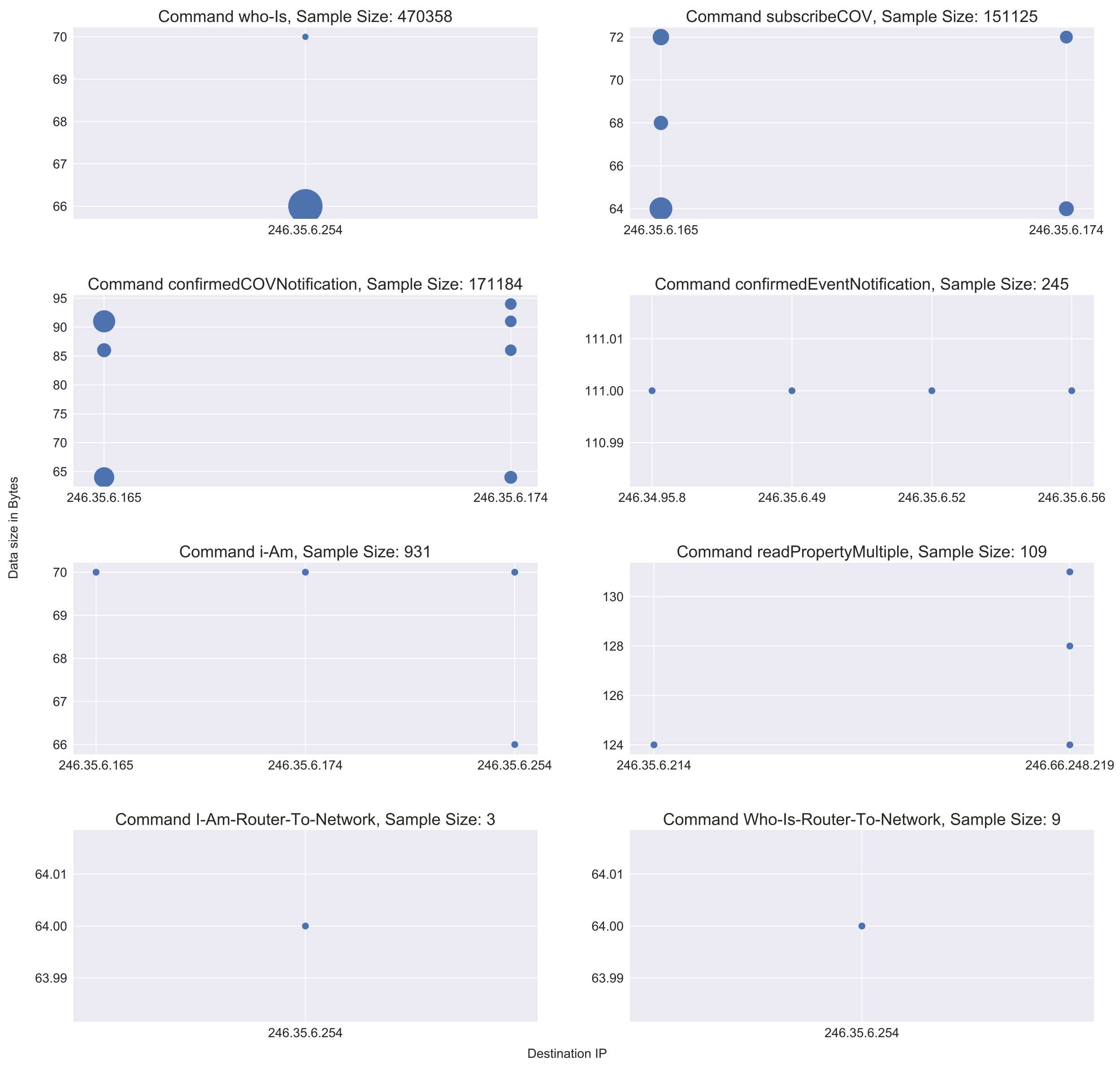


Figure 5.5: Scatter graph of packet size and destination for command packets sent by host 246.35.6.172

Table 5.3: Command breakdown of full network capture for the Real Normal Dataset

Command	Unique Hosts	Count	Total Command %
who-Is (Unconfirmed Service Request)	33	28,436,828	72.0318%
confirmedCOVNotification (SimpleAck)	5	4,222,241	10.6951%
confirmedCOVNotification (Confirmed Service Request)	6	4,154,232	10.5229%
i-Am (Unconfirmed Service Request)	47	881,131	2.2319%
subscribeCOV (Confirmed Service Request)	6	519,254	1.3153%
subscribeCOV (SimpleAck)	6	511,223	1.2950%
readProperty (Confirmed Service Request)	2	335,619	0.8501%
readProperty (ComplexAck)	1	334,488	0.8473%
writeProperty (Confirmed Service Request)	2	73,932	0.1873%
confirmedEventNotification (Confirmed Service Request)	2	4048	0.0103%
confirmedEventNotification (SimpleAck)	4	3294	0.0083%
ERROR:confirmedEventNotification	1	346	0.0009%
who-is-Router-to-Network (NPDU)	7	329	0.0008%
i-am-Router-to-Network (NPDU)	22	258	0.0007%
writeProperty (SimpleAck)	1	234	0.0006%
ICMP	3	192	0.0005%
readPropertyMultiple (ComplexAck)	2	167	0.0004%
readPropertyMultiple (Confirmed Service Request)	2	167	0.0004%
unconfirmedPrivateTransfer (Unconfirmed Service Request)	2	165	0.0004%

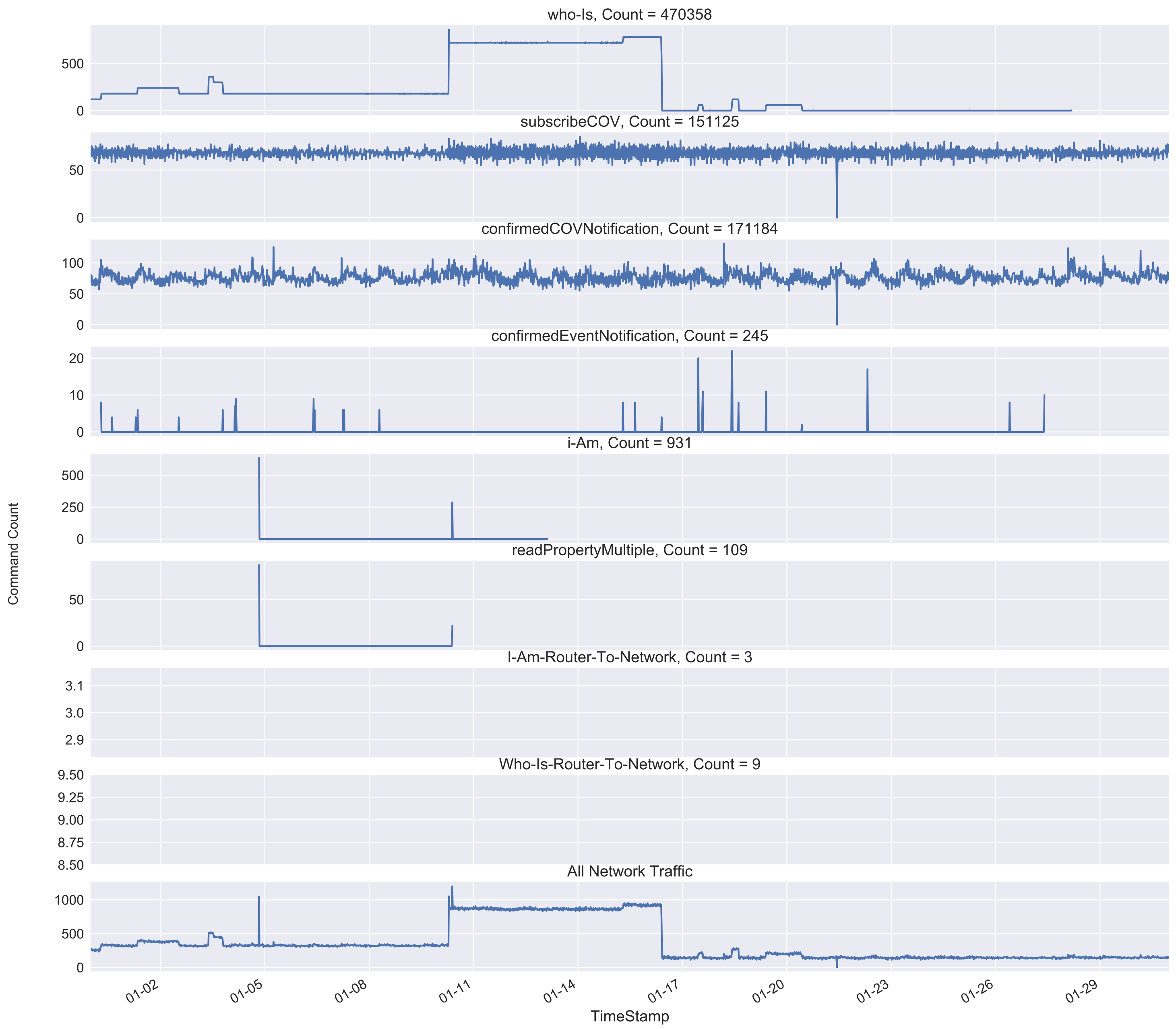


Figure 5.6: Command breakdown for host 246.35.6.172

So far, analysis has been concerned with the type of traffic transmitted across the network and the destination of that traffic. Time series analysis methods were explored to identify when traffic occurred. The primary aim for the time series analysis was to identify periodicity of network traffic. Seasonal decomposition using the *Python statsmodel* implementation of a naïve Loess function (Cleveland, Cleveland, McRae & Terpenning, 1990) was undertaken with a rolling window mean of one hour (referred to further as bins). Seasonal decomposition for each host was undertaken, in addition to the the full network in Figure 5.7. For the full network decomposition there is a large drop in traffic on the 22/01/18. Further investigation revealed a power outage on the university campus during this period after which the traffic returned to its previous pattern. Seasonal decomposition does not handle missing values, therefore sporadic traffic can be identified through preprocessing the data into one hour bins. 29 hosts were identified as having sporadic network activity, defined for this network as having less than 1,000 network flows over the duration of the network traffic, or hosts which go six hours without a flow. Hosts 246.35.6.172, 246.35.6.57, 246.35.6.60, and 246.35.6.71 have over 1,000 flows, and infrequently have less than two flows per hour. Hosts 246.35.6.57, 246.35.6.60 and 246.35.6.71 were missing one, two and six hour bins respectively over the entire duration of the dataset. As such, time based interpolation was applied to generate the missing bin values. For example, host 246.35.6.172 was missing values for 231 of 744 bins. However, on inspection 963 flows occur before 10:00/17/01/2018, after which 218 flows occur. This causes a large discrepancy in averaging methods, with one missed value before 17/01/2018, and 230 after this period. The periodic traffic changes volume, thus making interpolation over the entire set not truly explanatory. Therefore, the traffic was split into weekly records further analysis.

The mean weekly traffic for the network is outlined in Figure 5.8, which details the diurnal pattern of the network traffic, (the First of January in 2018 was a Monday). From the collection of weekly graphs, not reproduced here, some of the periodic hosts exhibit clear diurnal network patterns, while others have a static consistent packet count. Sporadic hosts are also clearly identified. In total, 22 hosts are classified as periodic. Of note, host 246.35.6.172 has two distinct behaviours, which prompted for further investigation of factors. A larger bin size of four hours was used for seasonal decomposition, reducing the amount of interpolation required from 230 to 34. Compared to the full dataset where the power outage can be seen, with traffic then returning to the regular pattern, it is unclear why the phase change occurs in host 246.35.6.172. It is clearly not a power outage, as the traffic does not return to a regular pattern, a larger network capture consisting of multiple months may reveal further information. Naïve analysis of the network has revealed interesting features for the Real Normal Dataset, which was applied to the synthetic datasets to identify differences in behaviour between normal and malicious actions.

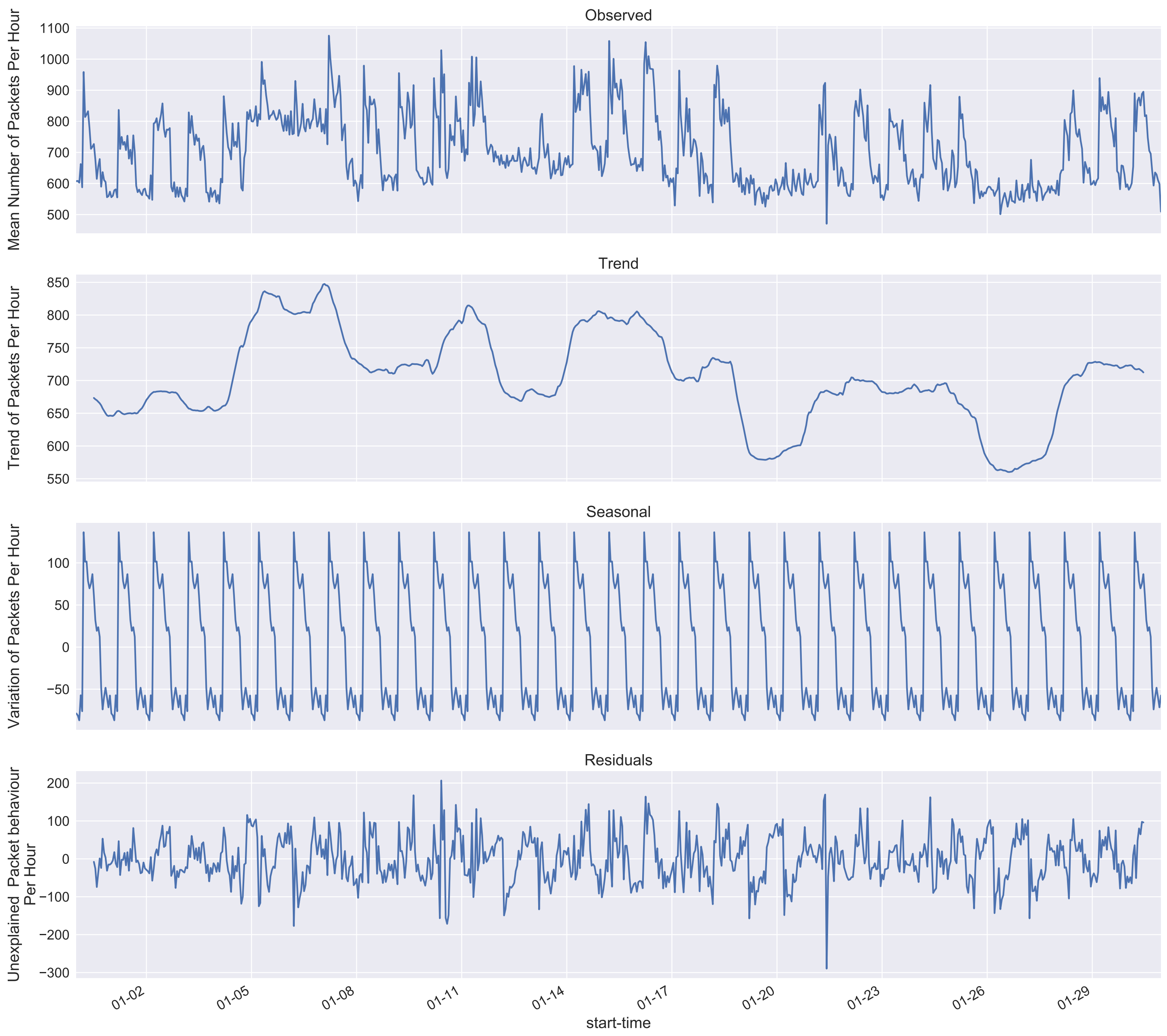


Figure 5.7: Seasonal decomposition of the Real Normal Dataset

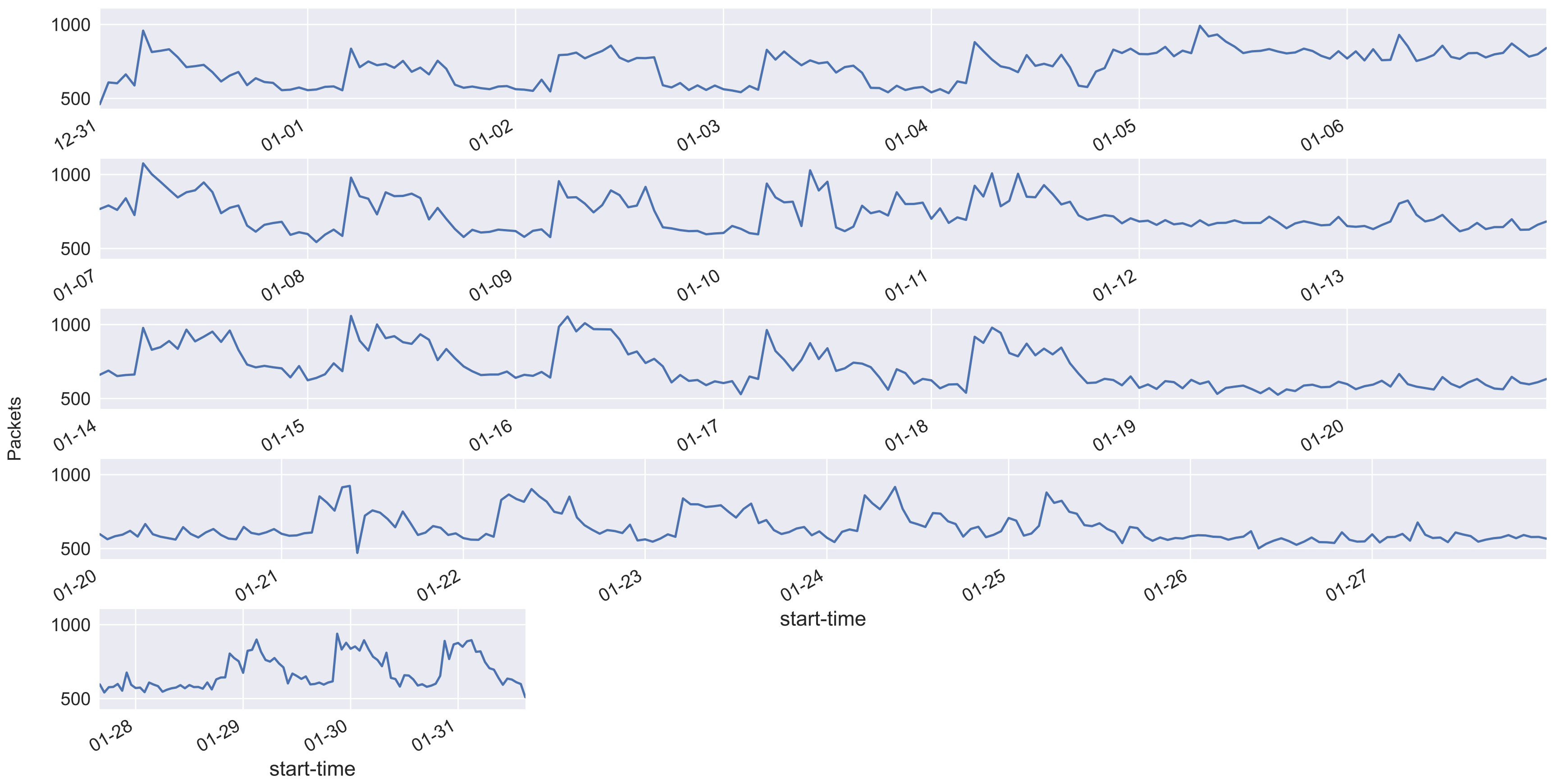


Figure 5.8: Average weekly time plot of the Real Normal Dataset

5.3 Synthetic Normal Dataset

This section presents the community detection, clustering, and naïve time series analysis for the Synthetic Normal Dataset. The Synthetic Normal Dataset consists of the network traffic from one month of the simulator defined in §4.6.5. Through initial exploration of the command sequences in the Synthetic Normal Dataset, 6 unexpected data troughs were identified. It was unclear what caused these reductions in traffic with no distinct pattern identified between occurrences. As this was not controlled behaviour, the data generation procedure was investigated, with the data in these windows regenerated to test for repeatability. Regenerating the data at these points did not replicate the unexpected troughs and may be an unexplained remnant of limitations in the base hardware implementation of ESXi. Given that the data affected represents 0.8% (six hours) of total traffic, investigating the cause of these unexpected reductions in traffic was deemed out of scope of the research. As such, the data for all hosts encompassing these points, totalling 206,995 frames were removed from the dataset.

5.3.1 Synthetic Normal Dataset: Analysis

The process described in §5.2.1 was also undertaken for the Synthetic Normal Dataset. The eigenvector centrality for the dataset describes the hosts which communicate the most with other hosts in the network. Details of the community detection algorithm applied to a sample of the hosts is outlined in Table 5.4. As the logger device at 192.168.10.91 consistently communicates with all other known devices in the network, it has the highest degree, and thus an eigenvector centrality of 1.0. The controllers are the second most connected, followed by the broadcast network address. While the broadcast address of 255.255.255.255 is clearly not a host, it is the destination address of 59 services in the network, as denoted by the Indegree value in Table 5.4, and as such has a measure of centrality in the network. Further, as detailed in §5.4.1, the broadcast network address is used in many BACnetwork attacks. Following the broadcast address, the thermostats and VAVs are the next most connected. Finally, the sensors and actuators, which normally only communicate with the managing controller, and the logging device are the least connected.

Six classes are defined with the Louvain algorithm, with a modularity ranking of 0.321 using the standard modularity resolution of 1.0. One class describes the controller, and all its connected sensors and actuators, in addition to the logger device, and one thermostat/VAV pair. A further three thermostat/VAV pairings are classed separately. The next class describes a controller, its sensors and the two remaining thermostat/VAV pairs. Finally, a class describes a controller, its sensors and actuators, the broadcast address and some sensors from a different controller. There are differences in the communication pathways between hosts from the design. It is of interest, given each grouping uses the same data, but resolves to different classes. Reducing the resolution of the Louvain algorithm increases the number of classes present in the network, as such a range of resolution values were enumerated. Reducing the modularity resolution to 0.8 resulted in nine classes with a modularity ranking of 0.323, where each thermostat/VAV pair is classed together, the two sets of controllers and actuators are classed together, one set of controllers, actuators and sensors are classed with the logger device, and the broadcast address and remaining sensors are classed together. For the Synthetic Normal Dataset, maximising the modularity ranking value presented a classification of nodes which was closer to the topological design of the simulator network. The network graph for the Normal Synthetic Dataset traffic at the maximised modularity ranking value is detailed in Figure 5.9.

Table 5.4: Hosts exhibiting measures of centrality, modularity and clustering coefficients for the Synthetic Normal Dataset

IP Address	Eigenvector centrality	Indegree	Outdegree	Degree	Modularity Class	Clustering Coefficient
192.168.10.91	1	118	112	230	6	0.12662
192.168.10.102	0.558824	51	59	110	4	0.405367
192.168.10.100	0.555273	52	56	108	6	0.409702
192.168.10.101	0.552736	50	58	108	5	0.414096
255.255.255.255	0.223583	59	0	59	2	0.064874
192.168.10.190	0.124058	12	15	27	0	0.295238
192.168.10.130	0.123843	12	16	28	1	0.279167
192.168.10.195	0.123101	12	15	27	4	0.295238
192.168.10.160	0.122332	12	13	25	5	0.217949
192.168.10.165	0.122332	12	13	25	3	0.217949
192.168.10.131	0.121563	11	11	22	1	0.44697
192.168.10.191	0.121561	11	11	22	0	0.44697
192.168.10.136	0.117654	11	11	22	7	0.44697
192.168.10.196	0.117407	10	11	21	4	0.472727
192.168.10.166	0.117255	10	11	21	3	0.472727
192.168.10.161	0.117255	10	11	21	5	0.472727
192.168.10.135	0.106489	11	16	27	7	0.279167
192.168.10.186	0.096857	5	5	10	4	0.433333
192.168.10.185	0.096857	5	5	10	4	0.433333
192.168.10.184	0.096857	5	5	10	4	0.433333

Flow records were generated for the Synthetic Normal Dataset, using the settings described in §5.1. 25,634,367 frames were processed into 406,426 flows for the 58 communicating hosts, see Table 5.5 for an overview. 42.1% of these flows (171,101) were uni-directional flows. 48,521 flows reached the Active-Flow-Timeout value, while 357,754 flows ended due to the Idle-Flow-Timeout being reached. The last 151 flows of the capture period ended prematurely, due to the end of the capture period being reached. Compared to the real network dataset there are far less active timeout flows (11.94% vs 55.75%), primarily due to the lack of constant *Who-Is* commands from the previously identified misconfigured device. Additionally, the inclusion of the logger device which cycles through each device in the network and reads specific object's properties causes a large number of idle flows. Specifically, as only one property is read for sensor devices, a large portion of network flows contain only two frames.

The command details for the full network of the Normal Synthetic Dataset are outlined in Table 5.6. Of note, 95.6379% of the network traffic consists of polling values from the sensors by the

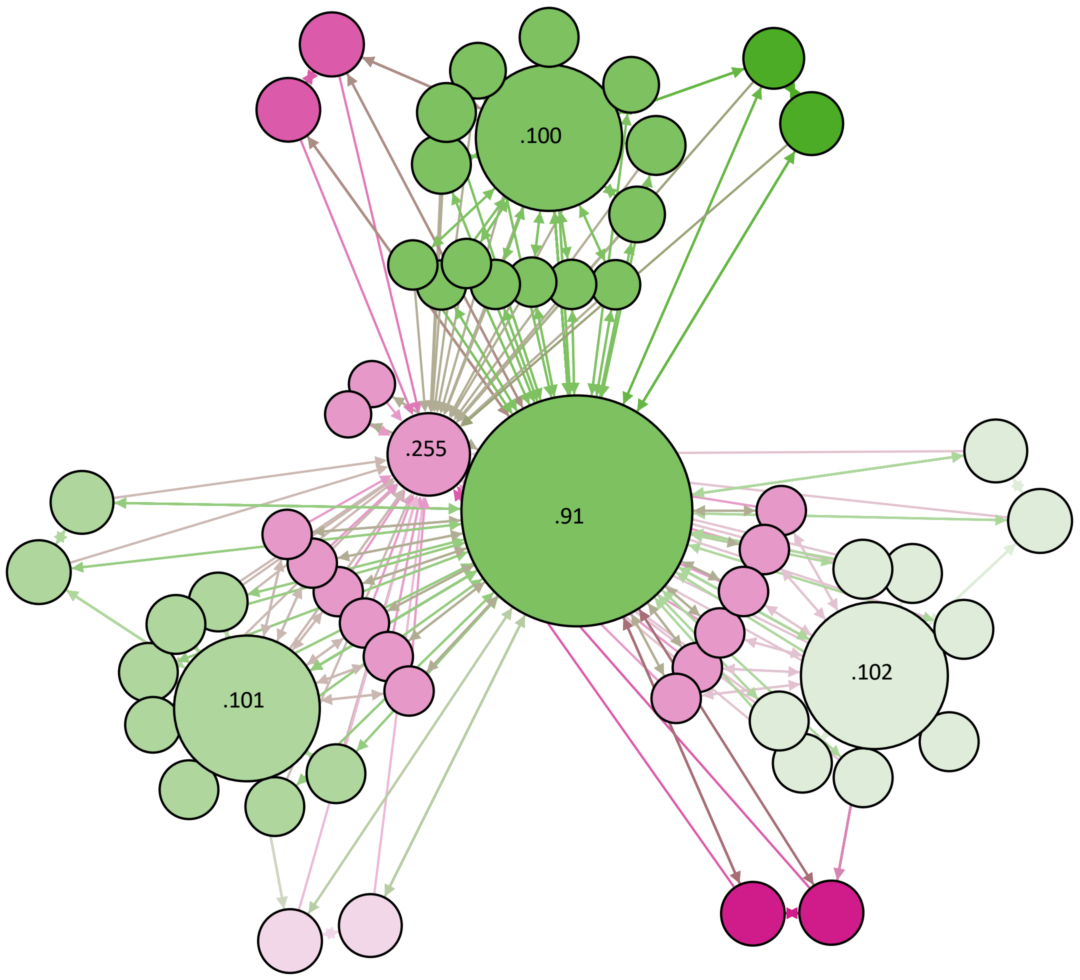


Figure 5.9: Synthetic Normal Dataset directed network graph

controller, thermostat and logger devices, via *Read property* requests and acknowledgements. This breakdown is quite different to the Real Normal Dataset, primarily due to the Real Normal Dataset only encompassing controller devices. The Synthetic Normal Dataset has primarily sensor and actuator devices, which are often not capable of CoV reporting, and thus have polling implemented using the *Read property* command. A scatter graph of the destination, and byte size of the *Write property* command for host 192.168.10.101 is shown in Figure 5.10. As detailed, the byte size of each command packet is identical across the hosts, due to a low variance in requests, and a similarly sized addressing scheme. If the addressing scheme for communicating devices differed, the byte size of the *Write property* command to each host may differ, or be unique per host such as in the Real Normal Dataset in Figure 5.5. The full breakdown of commands for the duration of the simulation for host 192.168.10.101 (a controller) is detailed in Figure 5.11.

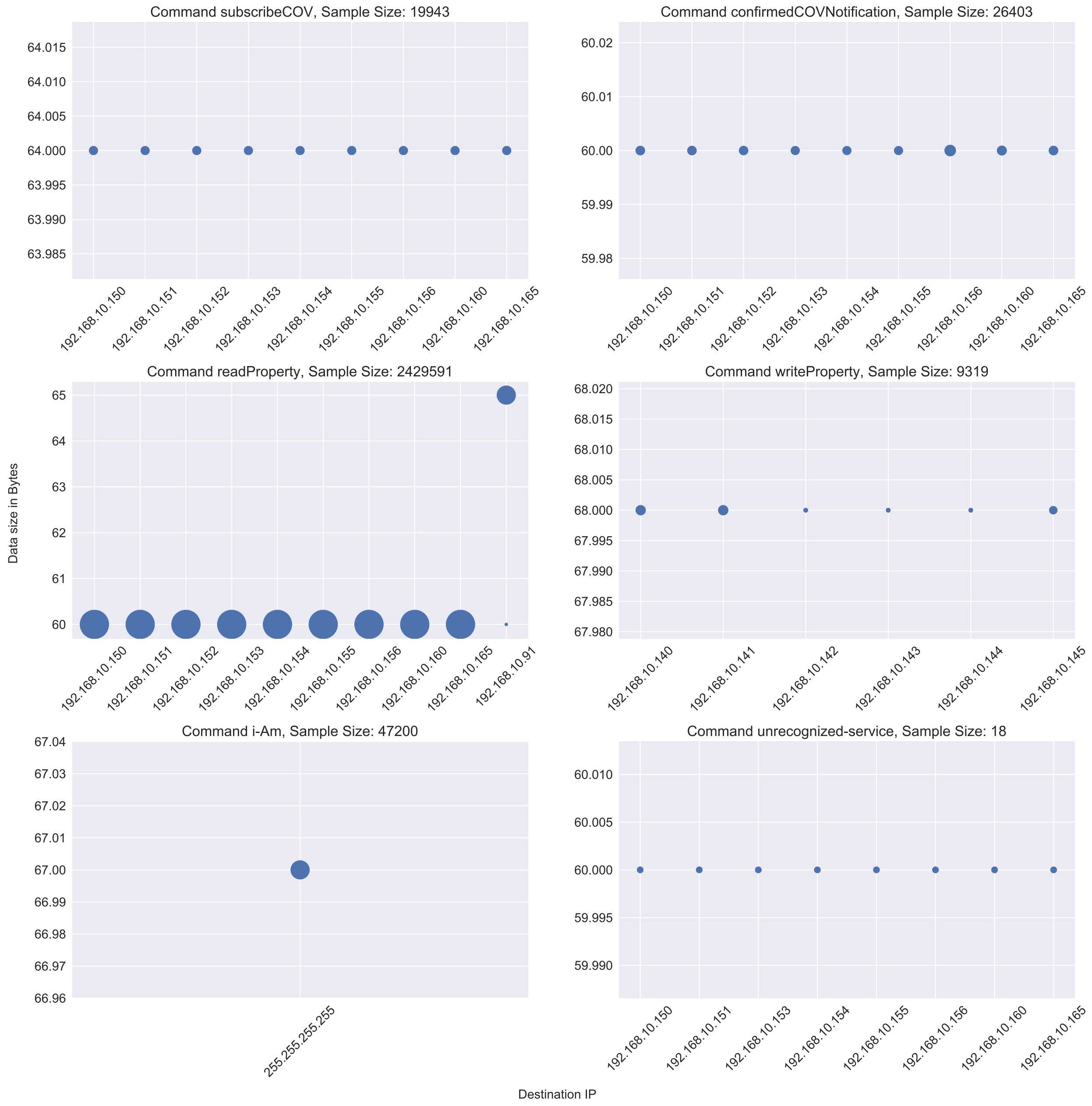


Figure 5.10: Scatter graph of packet size and destination for command packets sent by host 192.168.10.101

Table 5.5: Outline of generated flows and packet counts for a range of hosts in the Synthetic Normal Dataset scenario

Host	Flows	Packets	Total Flow %	Total Packet %
192.168.10.91	171100	1009053	42.0987%	3.9363%
192.168.10.100	32322	4911135	7.9527%	19.1584%
192.168.10.101	17399	4915843	4.2810%	19.1768%
192.168.10.102	17395	4913097	4.2800%	19.1661%
192.168.10.130	4430	1616197	1.0900%	6.3048%
192.168.10.160	4430	1616027	1.0900%	6.3041%
192.168.10.190	4429	1614971	1.0897%	6.3000%
192.168.10.135	4428	1614377	1.0895%	6.2977%
192.168.10.165	4427	1614480	1.0893%	6.2981%
192.168.10.195	4426	1612874	1.0890%	6.2918%
192.168.10.186	2961	7166	0.7285%	0.0280%
192.168.10.156	2960	6752	0.7283%	0.0263%
192.168.10.181	2953	4044	0.7266%	0.0158%
192.168.10.150	2952	3687	0.7263%	0.0144%
192.168.10.191	2952	11026	0.7263%	0.0430%
192.168.10.196	2952	11030	0.7263%	0.0430%
192.168.10.185	2951	3314	0.7261%	0.0129%
192.168.10.182	2951	3316	0.7261%	0.0129%
192.168.10.180	2951	3320	0.7261%	0.0130%
192.168.10.161	2951	9940	0.7261%	0.0388%
192.168.10.152	2951	3316	0.7261%	0.0129%
192.168.10.151	2951	3316	0.7261%	0.0129%
192.168.10.112	2951	2952	0.7261%	0.0115%
192.168.10.113	2951	2952	0.7261%	0.0115%
192.168.10.136	2951	9942	0.7261%	0.0388%
192.168.10.131	2951	9940	0.7261%	0.0388%
Total	**406426**	**25634367**		

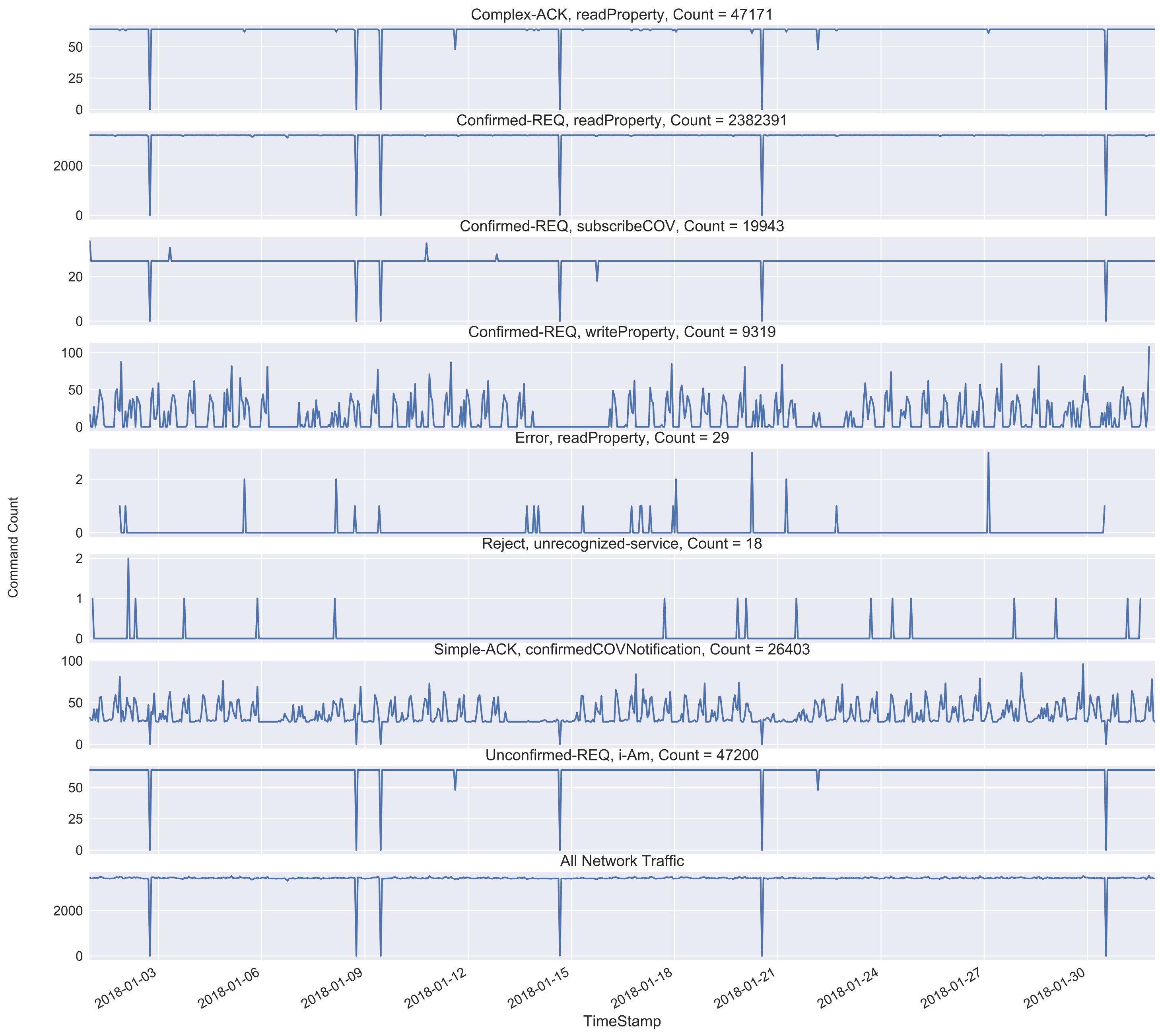


Figure 5.11: Command breakdown for host 192.168.10.101

Table 5.6: Command breakdown of full network capture for Synthetic Normal Dataset

Command	Unique Hosts	Count	Total Command %
readProperty(Confirmed-REQ)	10	12,258,404	47.8202%
readProperty(Complex-ACK)	57	12,257,753	47.8177%
i-Am(Unconfirmed-REQ)	57	336,302	1.3119%
who-Is(Unconfirmed-REQ)	1	336,299	1.3119%
confirmedCOVNotification(Confirmed-REQ)	33	104,569	0.4079%
confirmedCOVNotification(Simple-ACK)	9	104,459	0.4075%
subscribeCOV(Confirmed-REQ	9	86,440	0.3372%
subscribeCOV(Simple-ACK)	33	86,435	0.3372%
writeProperty(Confirmed-REQ	9	31,643	0.1234%
writeProperty(Simple-ACK)	24	31,643	0.1234%
readProperty(Error)	9	341	0.0013%
unrecognized-service(Reject)	7	79	0.0003%

A comparison of clustering approaches taken on the Synthetic Normal Dataset is detailed in Table 5.7. As the optimised clustering sizes increase in size, the start state tends to have a higher impact on the clustering, as can be seen when comparing the start state occurrences for the 1,000 selected state values. For example, the controllers and logger device (hosts 192.168.10.100, 192.168.10.101, 192.168.10.102, and 192.168.10.91, respectively) which optimise to over 25 states have low state occurrence rates and as such are heavily affected by the starting state for clustering each flow. It can be seen in Figure 5.12, where the AIC and BIC diverge in maximising the function that lowering the cluster size increases the occurrence of generating the same classifications from different starting states. Of interest, identified in Figure 5.12, many of the sensor and actuator hosts which contain multiple types of commands have reduced cluster sizes when applying the AIC and BIC. This implies the relationship between commands and cluster size is closer in the sensor and actuator devices than the controllers; perhaps due to the reduced variance in commands for sensors and actuators compared to controllers. A non-global optimums, or evaluating further cluster sizes with the AIC and BIC algorithms may reveal a better fitting cluster model for the higher interactive hosts, such as the controllers and logger. Additionally, using a simple max occurrences selection criteria does not necessarily identify the most appropriate start state, when evaluating using 1,000 differing start states. Some start states have similar occurrence rates, further enumeration of start state values may result in different starting state seed values when using an occurrence based selection criteria.

Due to the necessity of removing the unknown network data, interpolation was required for the seasonal decomposition of the network flows. Looking at the weekly breakdown of traffic is more descriptive in this case, (see Figure 5.16) highlighting the relatively steady state of the network during normal operation. The broken troughs in the network traffic are due to the six hours of removed traffic, outlined in §5.3. Further, the variation in packets per hour outlined in Figure 5.15, shows the cyclical nature of the traffic.

Table 5.7: Comparison of start states and occurrences with AIC/BIC selected cluster sizes for select Normal Synthetic Dataset hosts

IP	Naïve State	Naïve Occurrences	Cluster Size	AIC State	AIC Occurrences	AIC Cluster Size	BIC State	BIC Occurrences	BIC Cluster Size	Samples (flows)
192.168.10.100	0	492	8	277	1	27	277	1	27	32322
192.168.10.101	0	795	8	167	2	30	167	2	30	17399
192.168.10.102	1	369	8	397	3	30	397	3	30	17395
192.168.10.110	0	1000	5	0	1000	1	0	1000	1	2950
192.168.10.111	0	1000	5	0	1000	1	0	1000	1	2950
192.168.10.112	0	1000	5	0	1000	2	0	1000	2	2951
192.168.10.113	0	1000	5	0	1000	2	0	1000	2	2951
192.168.10.114	0	1000	5	0	1000	1	0	1000	1	2950
192.168.10.115	0	1000	5	0	1000	1	0	1000	1	2950
192.168.10.116	0	1000	4	0	1000	1	0	1000	1	2950
192.168.10.120	0	1000	2	0	1000	1	0	1000	1	2950
192.168.10.121	0	1000	2	0	1000	1	0	1000	1	2950
192.168.10.122	0	1000	2	0	1000	1	0	1000	1	2950
192.168.10.123	0	1000	2	0	1000	1	0	1000	1	2950
192.168.10.124	0	1000	2	0	1000	1	0	1000	1	2950
192.168.10.125	0	1000	2	0	1000	1	0	1000	1	2950
192.168.10.126	0	1000	2	0	1000	1	0	1000	1	2950
192.168.10.130	0	999	10	14	29	30	13	144	22	4430
192.168.10.131	0	347	5	0	706	4	0	706	4	2951
192.168.10.135	0	986	10	21	26	30	1	23	29	4428
192.168.10.136	2	559	5	0	723	4	0	723	4	2951
192.168.10.140	0	1000	3	0	1000	1	0	1000	1	2950
192.168.10.141	0	1000	3	0	1000	1	0	1000	1	2950
192.168.10.142	0	1000	3	0	1000	1	0	1000	1	2950
192.168.10.143	0	1000	3	0	1000	1	0	1000	1	2950
192.168.10.144	0	1000	3	0	1000	1	0	1000	1	2950
192.168.10.145	0	1000	3	0	1000	1	0	1000	1	2950

continuation of Table 5.7

IP	Naïve State	Naïve Occurrences	Cluster Size	AIC State	AIC Occurrences	AIC Cluster Size	BIC State	BIC Occurrences	BIC Cluster Size	Samples (flows)
192.168.10.146	0	1000	2	0	1000	1	0	1000	1	2950
192.168.10.150	0	1000	4	0	1000	3	0	1000	2	2952
192.168.10.151	0	1000	4	0	1000	2	0	1000	2	2951
192.168.10.152	0	1000	4	0	1000	2	0	1000	2	2951
192.168.10.153	0	1000	4	0	1000	1	0	1000	1	2950
192.168.10.154	0	1000	4	0	1000	1	0	1000	1	2950
192.168.10.155	0	1000	4	0	1000	1	0	1000	1	2950
192.168.10.156	0	990	4	0	1000	11	0	1000	2	2960
192.168.10.160	1	720	9	6	48	30	6	48	30	4430
192.168.10.161	1	455	5	2	954	3	2	954	3	2951
192.168.10.165	4	284	9	83	31	29	83	31	29	4427
192.168.10.166	1	559	5	1	707	4	1	707	4	2950
192.168.10.190	0	929	10	18	74	30	18	74	30	4429
192.168.10.191	1	699	5	3	423	6	3	423	6	2952
192.168.10.195	3	694	10	39	48	30	23	60	28	4426
192.168.10.196	0	818	5	9	289	9	0	983	4	2952
192.168.10.91	0	1000	2	885	1	29	343	2	22	171100

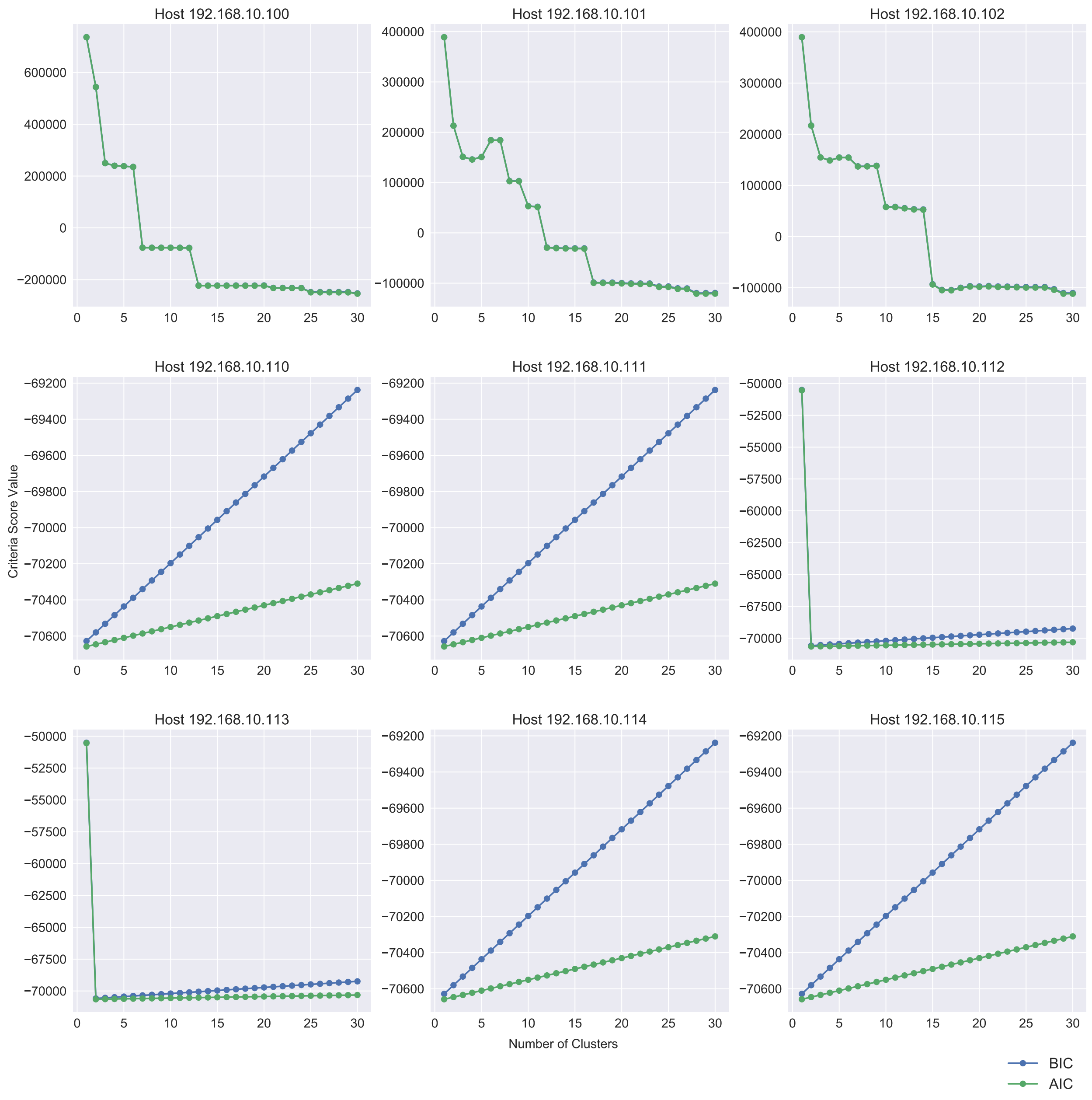


Figure 5.12: Comparison of AIC and BIC cluster size optimisations for a range of hosts in the Synthetic Normal Dataset

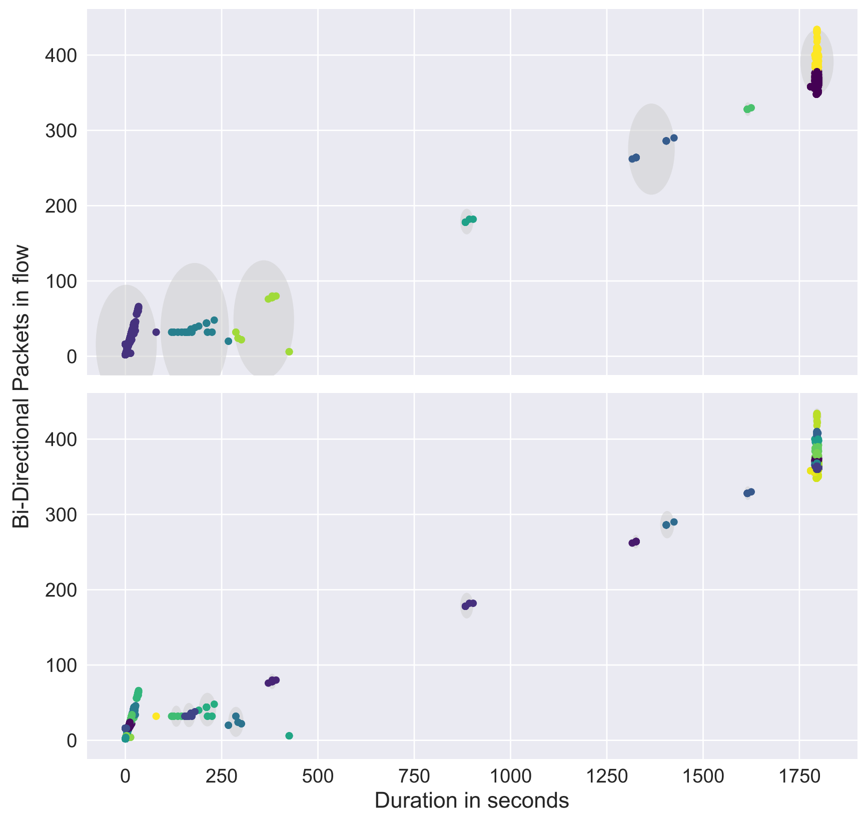


Figure 5.13: Comparison between naïve (k=8) and optimised (k=30) k-means clustering approach for host 192.168.10.101

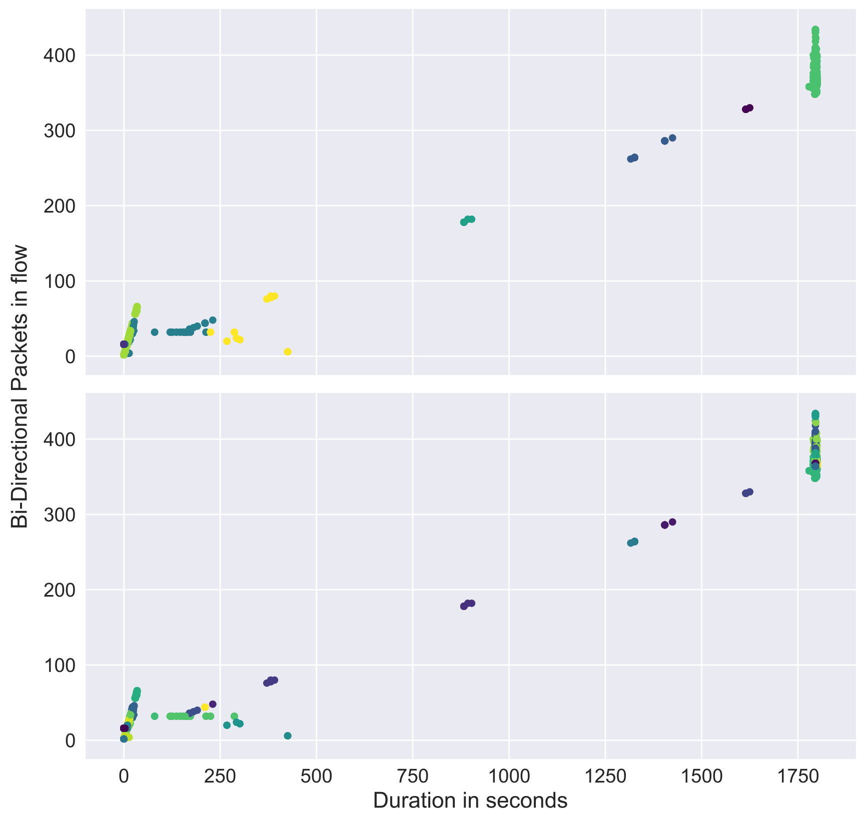


Figure 5.14: Comparison between naïve (k=8) and optimised (k=30) GMM clustering approach for host 192.168.10.101

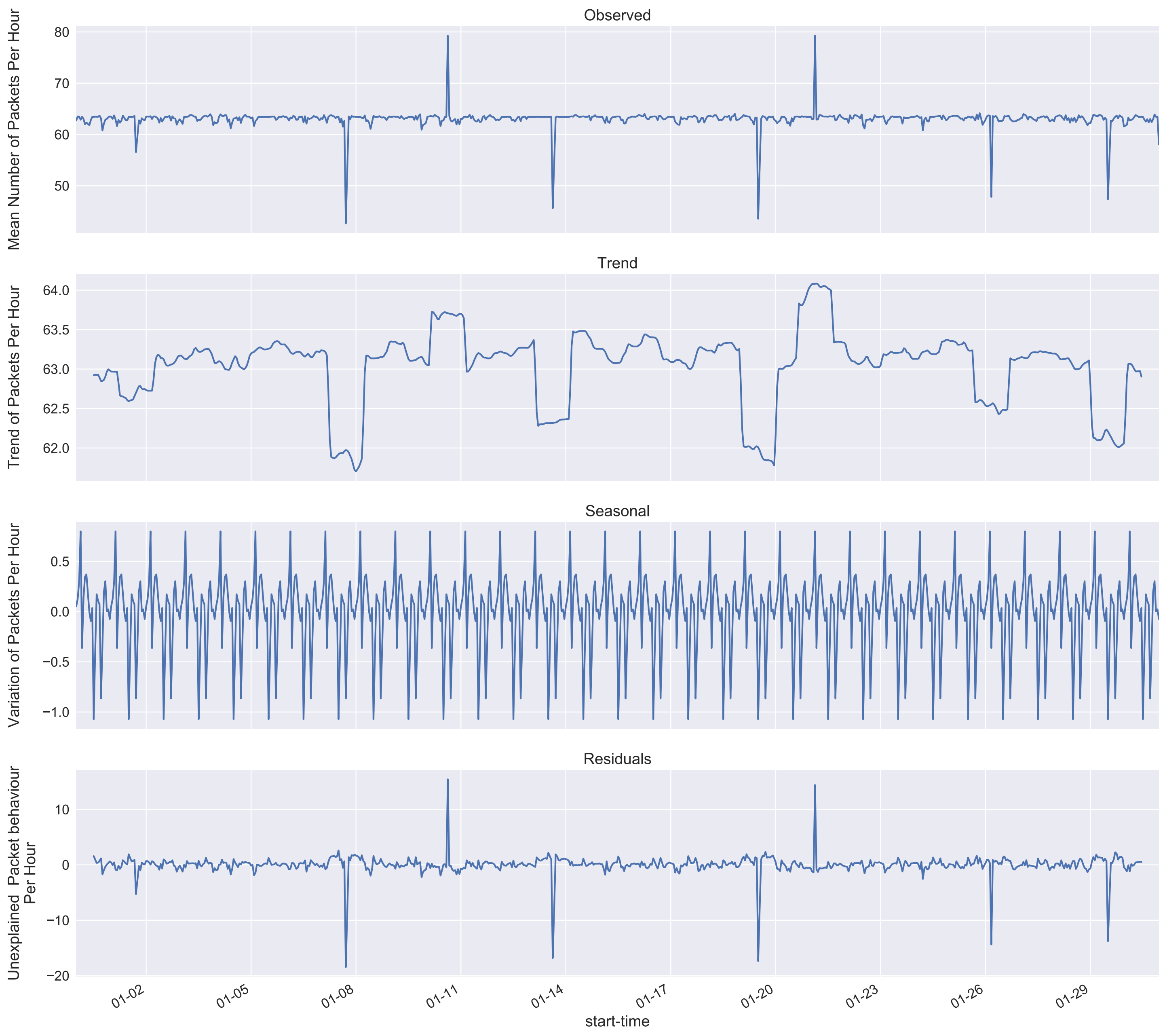


Figure 5.15: Seasonal decomposition of the Synthetic Normal Network

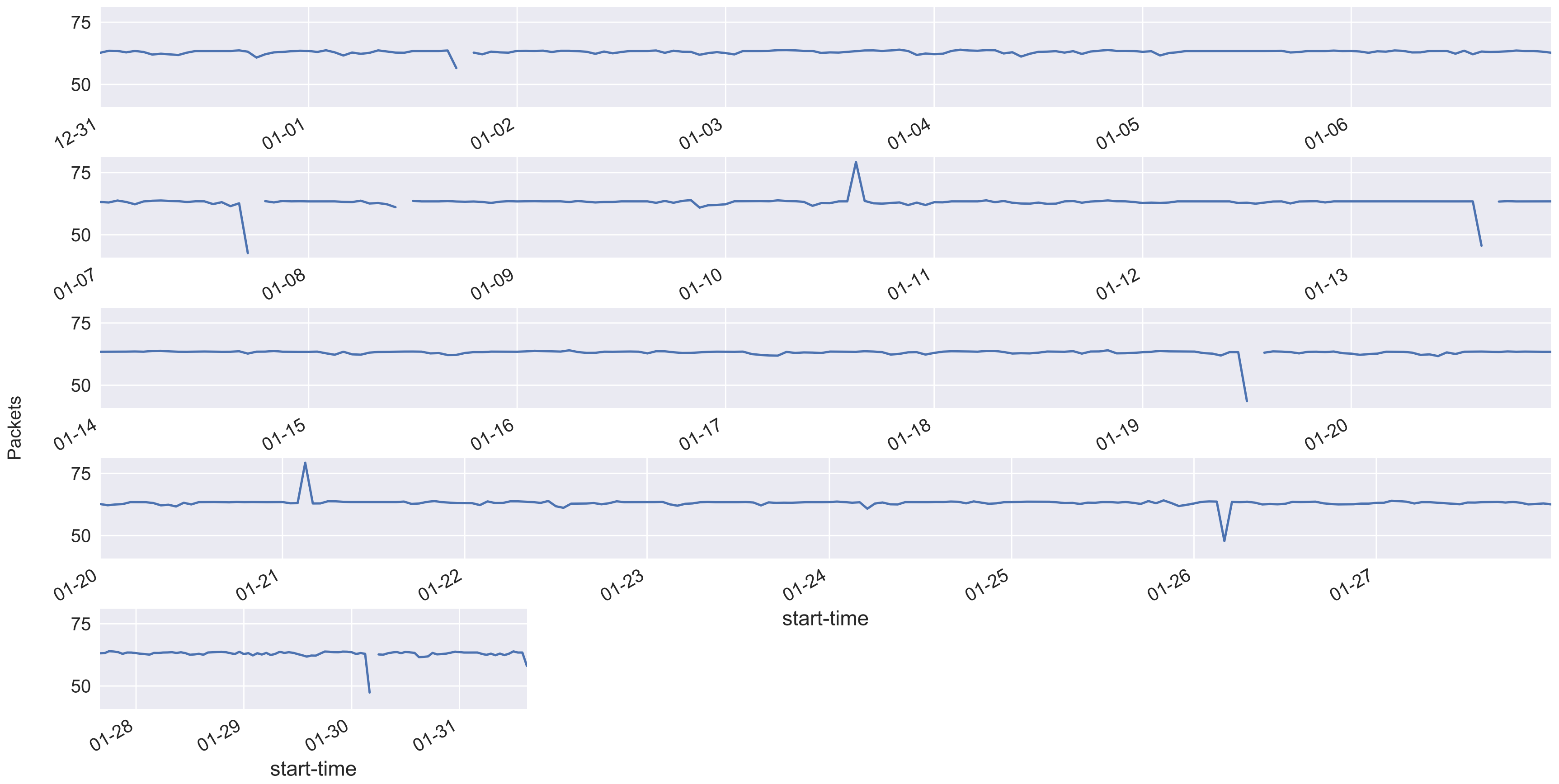


Figure 5.16: Average weekly packets for the Synthetic Normal Network

5.4 Synthetic Attack Dataset

The Synthetic Attack Dataset is constructed of a re-run of the first week of the network simulation with the attack framework defined in §4.6.9 implemented. In total, 163 attacks were launched against the simulation network over the one week duration from three adversarial hosts, one controller, one sensor and one external. The occurrence time, duration, and specific attack launched by each adversarial host were pseudo-randomly selected using the Python *random* function. The attack occurrences classed by attack and adversary type are outlined in Table 5.8.

Table 5.8: Ground truth attack occurrences in Synthetic Attack Dataset

Attack Class	Controller	Sensor	External
CovAttack	2	-	-
IamFlood	-	34	-
IamFloodSpoofIP	-	2	-
IamSpoofIP	-	5	-
PriorityArrayWrite	4	-	-
ReadPropertyAll	1	-	-
ReadPropSneak	3	-	-
RestartController	-	-	4
RestartDevice	-	-	18
WhoHasRecon	-	-	19
WhoIsFlood	8	4	-
WhoIsRecon	-	3	19
WhoIsReconSneak	2	1	-
WritePropInBounds	32	-	-
WritePropOutBounds	2	-	-
Total Per Device Type	**54**	**49**	**60**
Total			**163**

5.4.1 Synthetic Attack Dataset: Analysis

5,977,356 frames were processed into 93,755 flows for the 60 communicating hosts in the Synthetic Attack Dataset, see Table 5.10 for an overview. 42.68% of these flows (40,014) were uni-directional flows. 11,028 flows reached the Active-Flow-Timeout value, and 82,579 flows ended due to the Idle-Flow-Timeout being reached. The last 148 flows of the capture period ended prematurely, due to the end of the capture file being reached. Compared to the Synthetic Normal Dataset, there is a similar split for active (11.76% vs 11.94%) and idle (88.08% vs 88.03%) flow timeout's. Highlighting that cumulatively, the malicious commands in the Synthetic Attack Dataset do not cause large deviations in traffic type breakdown.

Six classes are identified with the Louvain algorithm with a modularity ranking of 0.179 using the standard modularity resolution of 1.0. Two classes are defined which represent individual thermostat/VAV pairings. Two further thermostat/VAV pairs are classed together. Two controllers and their associated sensors are classified as individual classes. Finally, a controller, the logging device, the external adversary, the broadcast address and the majority of actuators are in the final class. With a lower modularity ranking value the modularity resolution was reduced to investigate improved classification schemas. Optimising for the highest modularity resolution, a resolution of 0.9 or 0.8 can be used, where the modularity ranking increased to 0.228, with seven classes defined. The network graph for the Synthetic Attack dataset with modularity resolution 0.9 is presented in Figure 5.17. Comparing the network graph for the Synthetic Attack Dataset in Figure 5.17 to that of the Synthetic Normal Dataset in Figure 5.9, there are clear differences. The broadcast address becomes the central node to the network for the Synthetic Attack Dataset as outlined by the eigenvector centrality in Table 5.9. The increase can be attributed to the implemented network attacks as many of the attacks used the broadcast address to proliferate to all hosts. This is unlike both the Synthetic Normal and Real Normal Datasets, even with 75% of network traffic in the Real Normal Dataset directed at the broadcast address. When maximising the modularity ranking value, the number of classes reduces to seven from nine between the Synthetic Attack and Synthetic Normal graphs. It was expected that more classes would be defined when introducing more communication paths between hosts, however, the opposite occurred, with the malicious devices blurring the classes. Further, evident through visually comparing the two graphs, the Synthetic Attack Dataset graph is loosely clustered whereas the Synthetic Normal Dataset graph is tightly clustered. This suggests that the network attacks have had an effect on the underlying structure of the network, and thus presents a difference in the network pattern between normal and attack data.

The percentage breakdown of commands for the full Synthetic Attack Dataset is detailed in Table 5.11. There is little variation compared to the Synthetic Normal Dataset in Table 5.6 for the commands which were used in both datasets, due to the quantity of commands used in the attacks. To present individual results of the clustering and time series analysis the malicious controller host 192.168.10.101 was selected. The data size and destination for each command sent by host 192.168.10.101 is outlined in Figure 5.18. There is a variation in packet sizes between same destinations for some commands, unlike in the Synthetic Normal Dataset. This variation can be used as a distinguishing feature, however, it should be noted that the lack of size variance is due to the addressing scheme selected for the research. As can be seen in the Real Normal Dataset in Figure 5.5, where a higher variance in addressing exists, variance in packet size increases. However, packet size may still be a useful discriminator in networks with higher variance in packet sizes, with the caveat of a higher signal-to-noise ratio.

Evaluating Figure 5.19, compared to the Synthetic Normal Dataset in Figure 5.12, the optimised cluster sizes for all hosts are significantly larger, due to the wider range of messages passing over the network. This infers that clustering the flow traffic can distinguish between different BACnet commands sent over the network. However, a similar trend follows whereby the controller hosts with a high variance of commands have lower occurrence rates, regardless of sample size, outlined in Table 5.12. The AIC and BIC for sensor hosts (192.168.10.110-115 inclusive) in Figure 5.19 however suggests smaller sized clusters, and identify potential non-global optima when using larger cluster

Table 5.9: Hosts exhibiting measures of centrality, modularity and clustering coefficients for the Synthetic Attack Dataset

IP Address	Eigenvector centrality	Indegree	Outdegree	Degree	Modularity Class	Clustering Coefficient
255.255.255.255	1	402	0	402	5	0.087071
192.168.10.91	0.511762	115	162	277	5	0.125776
192.168.10.101	0.310557	58	115	173	3	0.332876
192.168.10.102	0.281355	51	72	123	2	0.434551
192.168.10.100	0.276146	51	71	122	5	0.42234
192.168.10.151	0.139735	15	8	23	3	0.316993
192.168.10.183	0.128872	14	8	22	3	0.330882
192.168.10.112	0.12853	14	14	28	3	0.34632
192.168.10.145	0.088325	9	11	20	3	0.367647
192.168.10.160	0.087015	14	18	32	0	0.345029
192.168.10.182	0.076697	8	14	22	2	0.380952
192.168.10.165	0.075681	13	17	30	1	0.356209
192.168.10.161	0.074783	11	14	25	0	0.47619
192.168.10.130	0.074618	13	18	31	5	0.356725
192.168.10.131	0.072662	11	14	25	5	0.47619
192.168.10.190	0.07255	13	17	30	2	0.356209
192.168.10.136	0.071255	11	14	25	4	0.47619
192.168.10.135	0.066711	12	17	29	4	0.356209
192.168.10.195	0.064528	12	19	31	2	0.356725
192.168.10.154	0.06262	6	7	13	3	0.472222
192.168.10.155	0.06262	6	7	13	3	0.472222
192.168.10.156	0.06262	6	7	13	3	0.472222
192.168.10.150	0.06262	6	7	13	3	0.444444
192.168.10.152	0.06262	6	7	13	3	0.472222
192.168.10.153	0.06262	6	7	13	3	0.472222
192.168.10.180	0.05956	6	7	13	2	0.472222
...	...	...	...	...	...	...
192.168.10.66	0.00895	2	51	53	5	0.038431

sizes, specifically for hosts 192.168.10.110 and 192.168.10.114. As expected, the BIC generally suggest simpler clustering models compared to the AIC. For host 192.168.10.101, the AIC and BIC both minimised to a 30 cluster size. As such, only the AIC is presented in Figures 5.20 and 5.21. It is difficult to evaluate the effectiveness of increasing the clustering size. For host 192.168.10.101 the error rate differential between the naïve and optimised cluster sizes (11 and 30) is 978.8 for the BIC and 1697.6 for the AIC from a cluster size increase of 19. This is not a large increase, and perhaps the optimised cluster size should not be selected. However, this is unlike host 192.168.10.100, where the error rate differential is 48729.3 for the BIC and 49557.7 for the AIC between the naïve and optimised cluster sizes (10 and 30). A different approach is required to identify the optimal cluster sizes for this dataset to reduce potential overfit. As can be identified in Figures 5.20 and 5.21, there seems to be little added benefit for understanding the breakdown of flows for using a 30 cluster size approach.

Figure 5.22 details the command breakdown over time for the malicious controller host 192.168.10.101. For the *Write property* command, a number of spikes can be identified which relate to frequency

Table 5.10: Outline of generated flows and packet counts for a range of hosts in the Synthetic Attack Dataset scenario

Host	Flows	Packets	Total Flow %	Total Packet %
192.168.10.91	38976	229857	41.57218%	3.84546%
192.168.10.100	7388	1119081	7.88011%	18.72201%
192.168.10.101	4063	1193926	4.33364%	19.97415%
192.168.10.102	4007	1121788	4.27391%	18.76729%
192.168.10.130	1025	367687	1.09328%	6.15133%
192.168.10.190	1025	368494	1.09328%	6.16483%
192.168.10.165	1024	367036	1.09221%	6.14044%
192.168.10.135	1024	367489	1.09221%	6.14802%
192.168.10.160	1024	368451	1.09221%	6.16411%
192.168.10.195	1023	367241	1.09114%	6.14387%
192.168.10.182	707	2731	0.75409%	0.04569%
192.168.10.145	698	23572	0.74449%	0.39435%
192.168.10.144	697	1489	0.74343%	0.02491%
...	...	...	...	...
192.168.10.66	87	1002	0.09280%	0.01676%
Total	**93755**	**5977356**		

Table 5.11: Command breakdown of the Synthetic Attack Dataset

Command	Unique Hosts	Count	Total Command %
readProperty (Confirmed-REQ)	10	2,790,724	46.69%
readProperty (Complex-ACK)	57	2,790,182	46.68%
i-Am (Unconfirmed-REQ)	57	137,961	2.31%
who-Is (Unconfirmed-REQ)	5	100,078	1.67%
writeProperty (Confirmed-REQ)	9	32,400	0.54%
writeProperty (Simple-ACK)	24	32,001	0.54%
confirmedCOVNotification (Confirmed-REQ)	33	23,750	0.40%
confirmedCOVNotification (Simple-ACK)	9	23,709	0.40%
subscribeCOV (Confirmed-REQ)	9	19,741	0.33%
subscribeCOV (Simple-ACK)	33	19,738	0.33%
i-Have (Unconfirmed-REQ)	57	5,216	0.09%
who-Has (Unconfirmed-REQ)	1	912	0.02%
readProperty (Error)	13	466	0.01%
writeProperty (Error)	2	399	0.01%
unrecognized-service (Reject)	3	31	0.00%
reinitializeDevice (Confirmed-REQ)	1	22	0.00%
reinitializeDevice (Simple-ACK)	2	22	0.00%
readPropertyMultiple (Complex-ACK)	1	1	0.00%
readPropertyMultiple (Confirmed-REQ)	1	1	0.00%

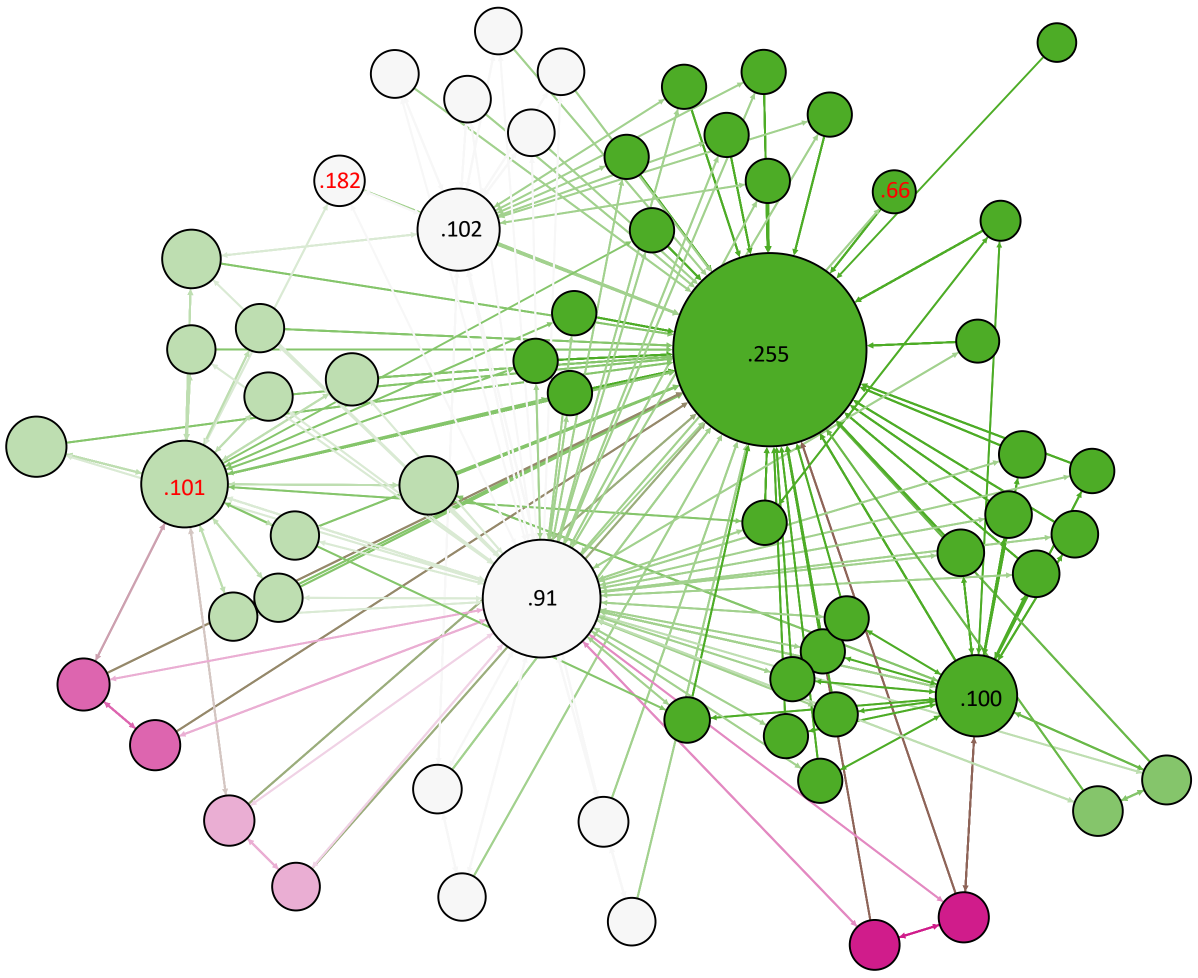


Figure 5.17: Synthetic Attack Dataset directed network graph, malicious hosts are identified in red

attacks launched by the controller against other hosts in the network. When compared to host 192.168.10.102 in Figure 5.23, a non-malicious controller, the *Write property* command difference is clear. For frequency-based attacks, as noted in §4.4.1, counts over time are a useful measure. The trend analysis in Figure 5.24 shows the corresponding peaks in network traffic, and the increasing trend of data until the network attacks finish. Comparing the trend analysis in Figure 5.24 to the first week of average weekly packets in the Synthetic Normal dataset in Figure 5.16, there are clear differences from the steady state.

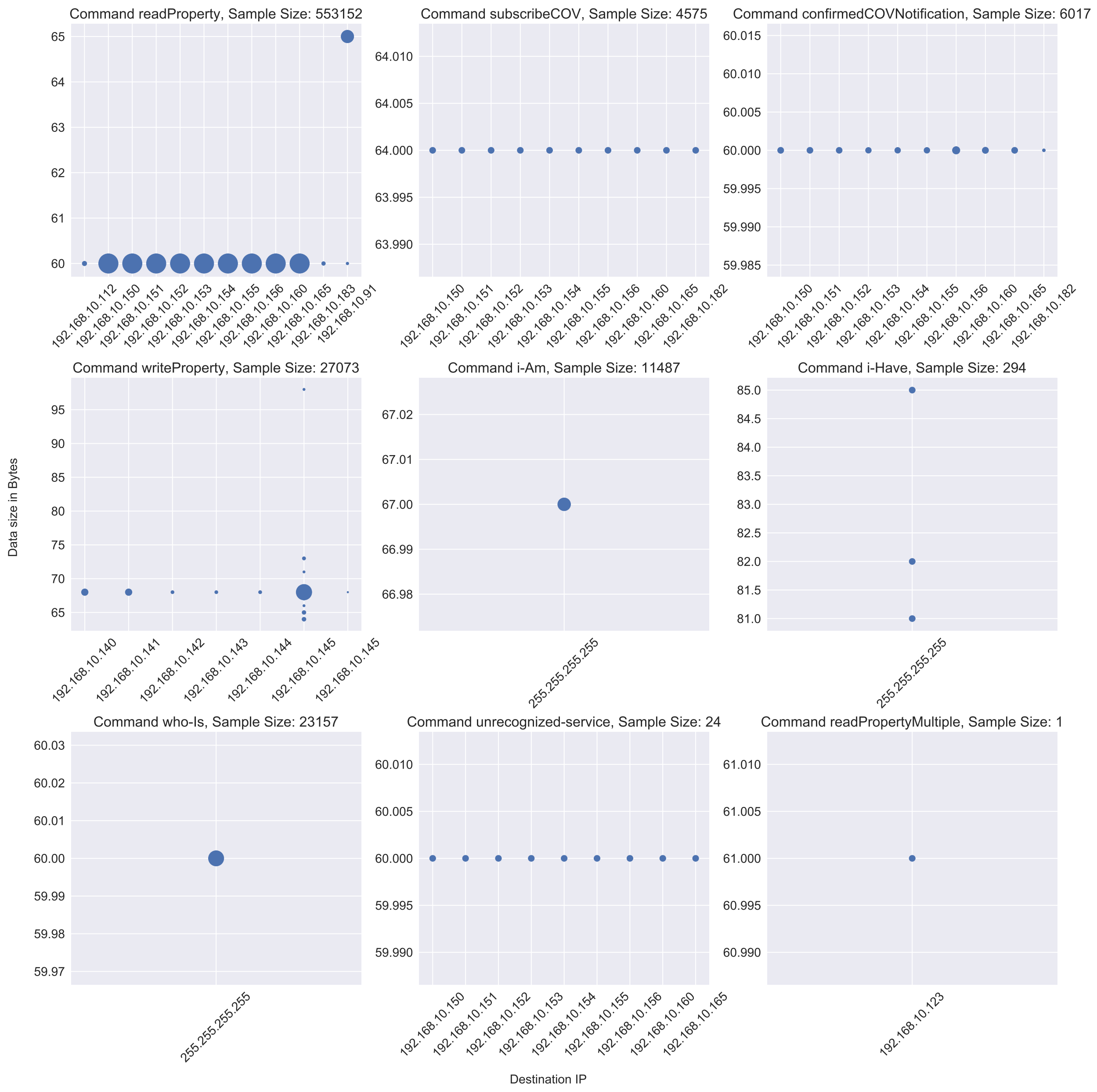


Figure 5.18: Scatter graph of packet size and destination for command packets sent by host 192.168.10.101

Table 5.12: Comparison of start states and occurrences with AIC/BIC selected cluster sizes for Synthetic Attack Dataset hosts

IP	Naïve State	Naïve Occurrences	Cluster Size	AIC State	AIC Occurrences	AIC Cluster Size	BIC State	BIC Occurrences	BIC Cluster Size	Samples (flows)
192.168.10.100	0	168	10	876	2	30	485	2	29	7388
192.168.10.101	11	249	11	598	2	30	598	2	30	4063
192.168.10.102	12	53	9	347	3	22	347	3	22	4007
192.168.10.110	3	508	6	3	219	21	1	444	9	689
192.168.10.111	0	921	6	0	665	30	0	901	15	689
192.168.10.112	1	486	7	0	246	15	0	246	15	687
192.168.10.113	0	265	6	0	785	30	10	201	10	688
192.168.10.114	0	786	6	0	501	23	0	573	14	689
192.168.10.115	1	517	6	0	877	30	2	352	11	688
192.168.10.116	2	477	5	0	934	30	0	934	30	689
192.168.10.120	0	1000	3	0	798	30	0	872	9	689
192.168.10.121	0	999	3	3	512	30	0	605	10	688
192.168.10.122	0	1000	3	0	802	30	0	845	9	688
192.168.10.123	0	927	4	1	587	30	1	465	10	689
192.168.10.124	0	1000	3	0	768	30	0	799	8	688
192.168.10.125	0	999	3	0	823	30	0	802	8	689
192.168.10.126	0	1000	3	0	822	30	6	298	9	689
192.168.10.130	0	568	10	0	411	19	3	170	17	1025
192.168.10.131	0	725	6	1	638	13	1	638	13	691
192.168.10.135	0	942	10	0	508	19	0	508	19	1024
192.168.10.136	2	420	6	0	621	11	0	621	11	690
192.168.10.140	0	989	4	0	767	30	4	244	11	688
192.168.10.141	0	997	4	4	293	30	2	392	13	689
192.168.10.142	2	369	4	0	632	14	0	632	14	687
192.168.10.143	0	988	4	0	859	30	0	875	28	688
192.168.10.144	0	299	5	3	308	12	3	308	12	697
192.168.10.145	0	999	5	1	271	25	1	271	25	698

continuation of Table 5.12

IP	Naïve State	Naïve Occurrences	Cluster Size	AIC State	AIC Occurrences	AIC Cluster Size	BIC State	BIC Occurrences	BIC Cluster Size	Samples (flows)
192.168.10.146	0	1000	3	0	658	30	0	734	13	689
192.168.10.150	0	556	5	0	976	30	1	483	12	690
192.168.10.151	2	431	6	0	973	30	0	497	12	687
192.168.10.152	1	514	5	0	920	30	0	911	9	688
192.168.10.153	0	626	5	0	969	30	0	291	12	689
192.168.10.154	3	265	5	0	710	30	1	635	10	688
192.168.10.155	0	260	5	0	951	30	0	612	10	688
192.168.10.156	0	988	5	0	912	30	0	543	14	690
192.168.10.160	0	547	11	0	854	21	1	358	16	1024
192.168.10.161	5	417	6	0	573	14	0	561	11	689
192.168.10.165	0	580	10	0	393	18	0	393	18	1024
192.168.10.166	1	266	6	0	796	16	0	589	11	689
192.168.10.180	0	463	5	0	924	30	0	858	9	689
192.168.10.181	1	315	5	0	897	30	1	687	13	688
192.168.10.182	0	366	7	26	37	17	26	37	17	707
192.168.10.183	0	579	6	3	568	30	0	810	10	688
192.168.10.184	0	615	5	2	830	30	2	458	13	689
192.168.10.185	2	235	5	0	958	30	1	673	9	688
192.168.10.186	4	383	5	0	929	30	0	902	9	687
192.168.10.190	1	447	10	0	297	23	4	376	17	1025
192.168.10.191	3	219	6	0	617	13	2	592	11	688
192.168.10.195	0	629	10	1	421	18	3	163	13	1023
192.168.10.196	1	405	6	1	865	16	1	865	16	690
192.168.10.66	0	1000	3	0	1000	5	0	1000	4	87
192.168.10.91	0	1000	2	469	2	25	0	808	10	38976

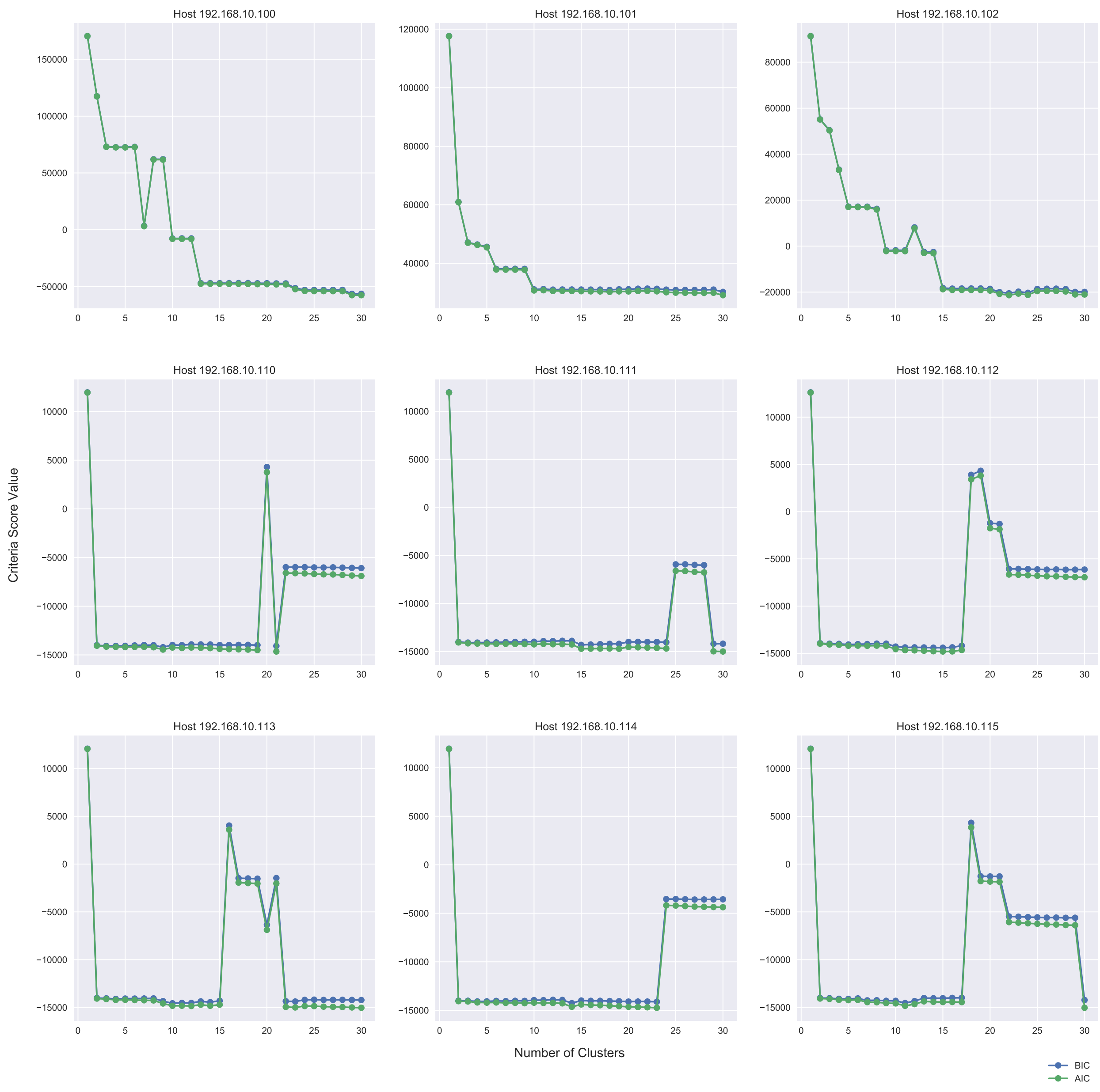


Figure 5.19: Comparison of AIC and BIC cluster size optimisations for a range of hosts in the Synthetic Attack Dataset

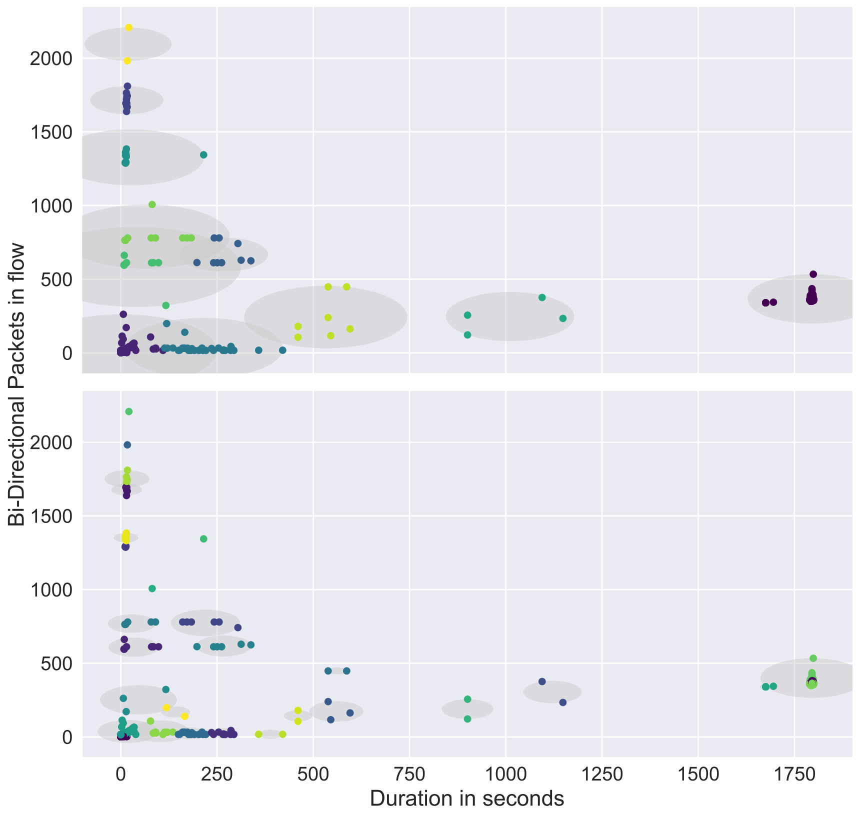


Figure 5.20: Comparison between naïve (k=11) and optimised (k=30) GMM clustering approach for host 192.168.10.101

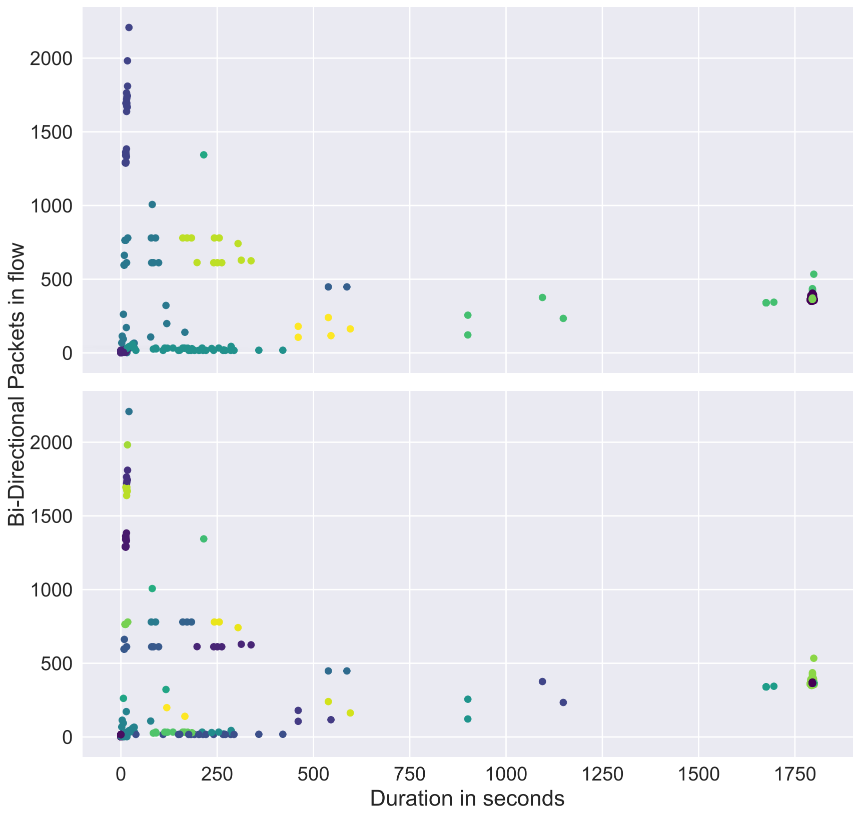


Figure 5.21: Comparison between naïve (k=11) and optimised (k=30) GMM clustering approach for host 192.168.10.101

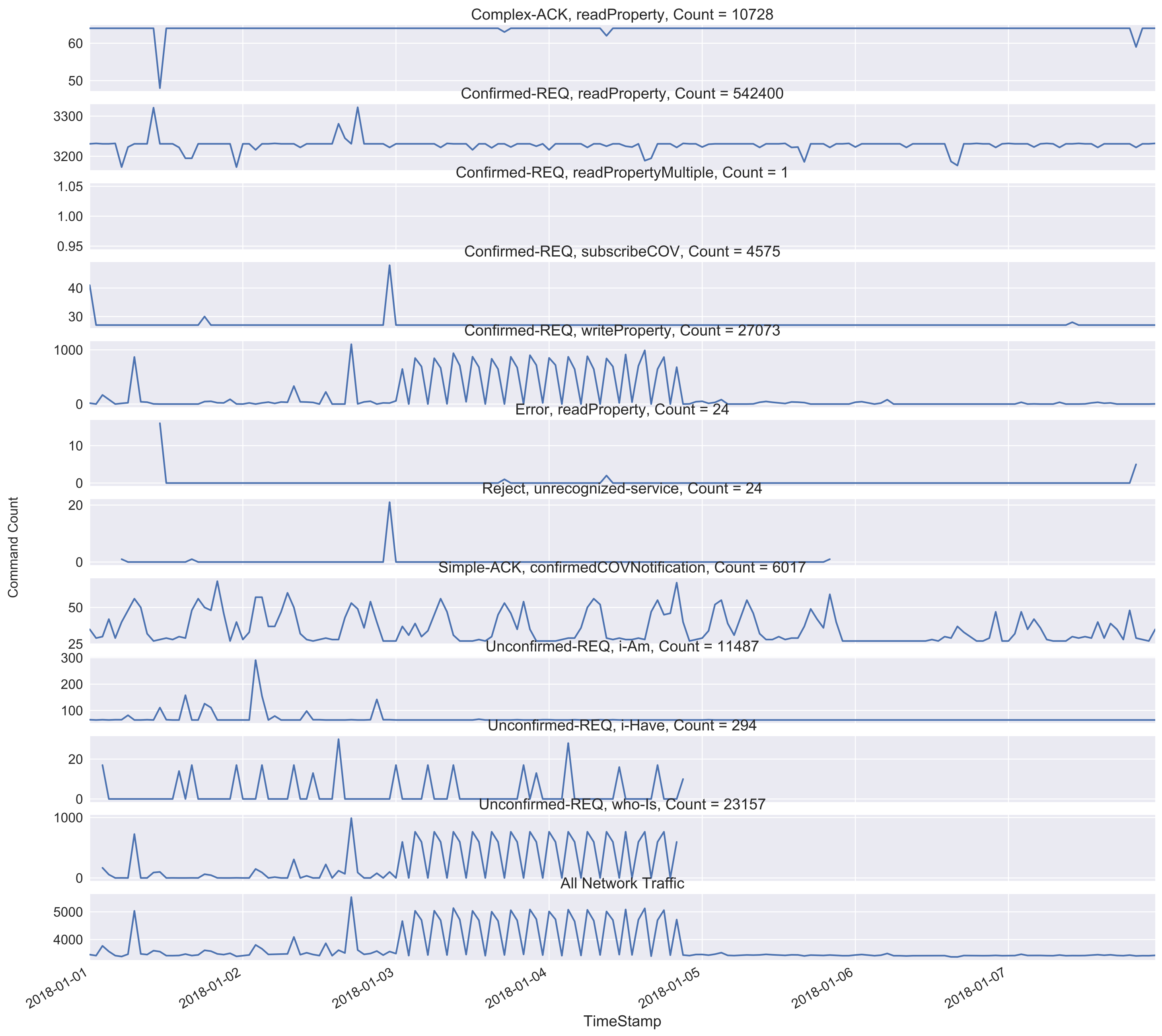


Figure 5.22: Command breakdown for Host 192.168.10.101

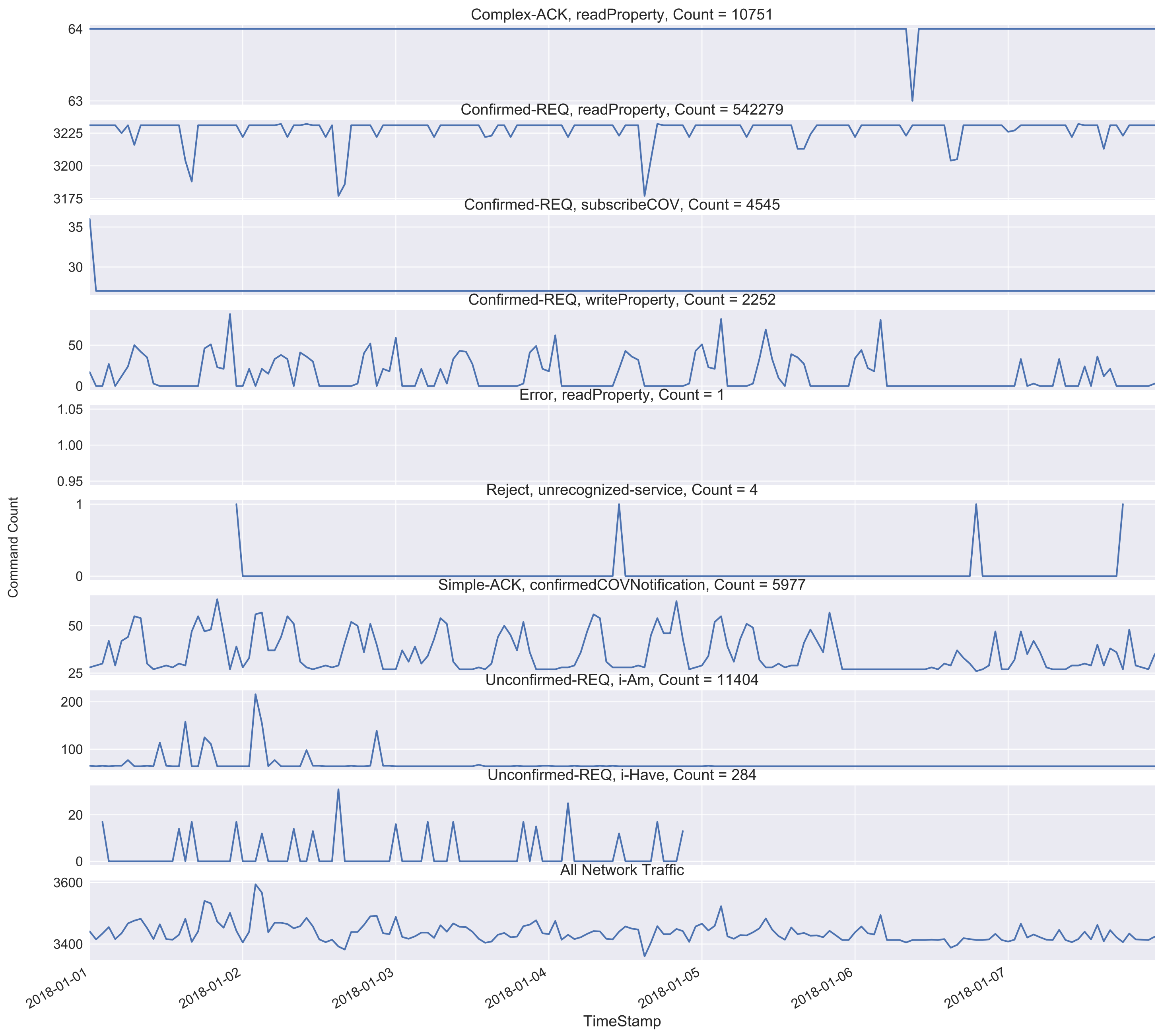


Figure 5.23: Command breakdown for Host 192.168.10.102

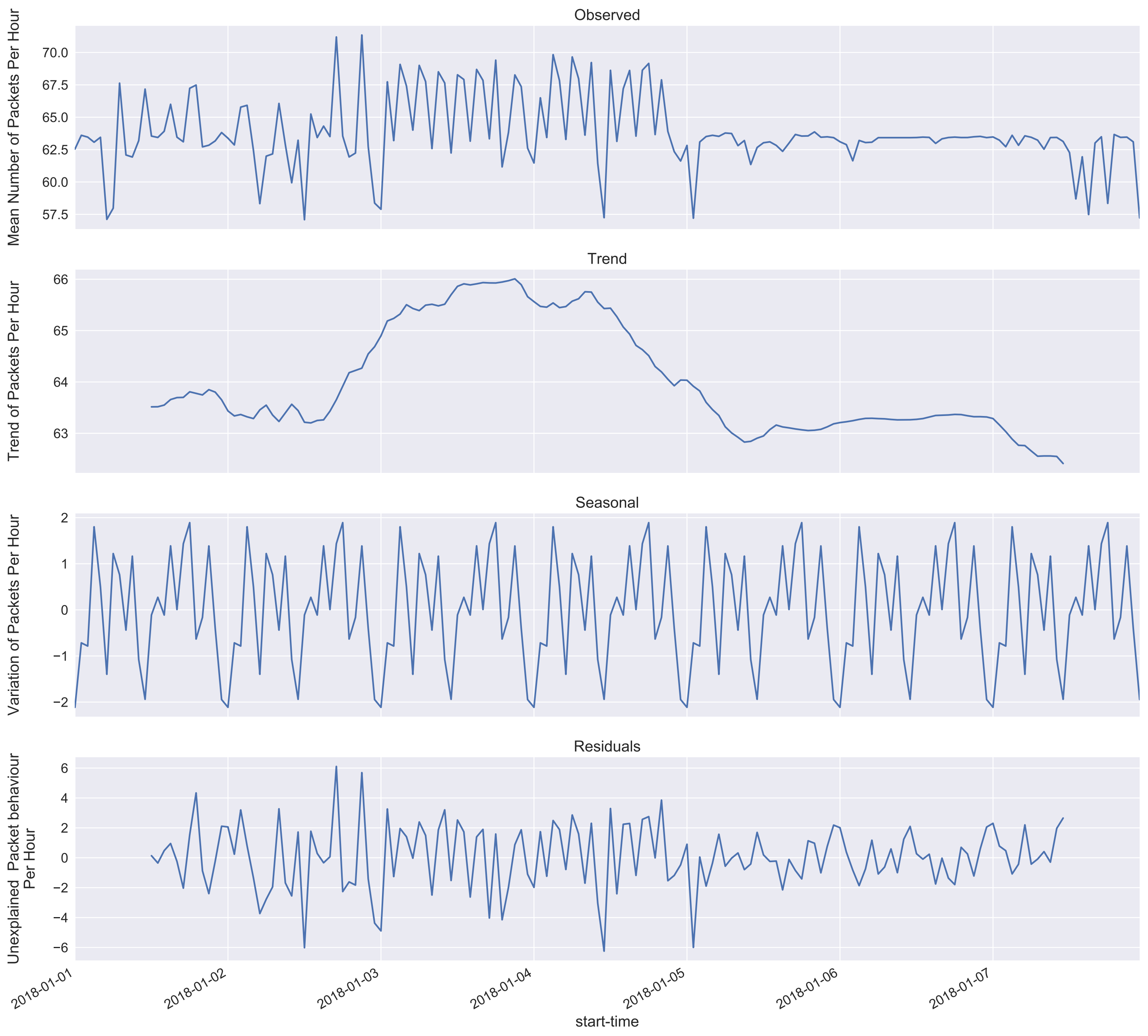


Figure 5.24: Seasonal decomposition of the Synthetic Attack Dataset

5.4.2 Synthetic Dataset comparisons

While comparing the two dataset clusters side-by-side distinctly reveals many of the network attacks, this is not a realistic approach for real network detection. Without a ground truth, as would be faced when applying exploratory analysis to real network data, the differences between legitimate and malicious traffic have limited contextual meaning.

From this unsupervised exploration of the Synthetic Normal and Synthetic Attack Datasets, known BACnet attacks have a clearly distinguishable network pattern. For the simulated data, many of the commands do not occur normally, and as such can be identified when used for a malicious purpose. Comparing the use of these identifiable commands to the command breakdown of the Real Normal Dataset, commands such as *Who-has*, and *I-have*, are low in quantity, or non-existent. Thus, for these types of commands, existence is enough for identification and investigation. Other core commands, such as *Read property*, *Write property*, *I-am* and *Who-is* can be identified by distinguishable patterns based on frequency, or variable packet sizes. These patterns are highlighted through the difference in cluster structure between Synthetic Normal and Synthetic Attack results in Figures 5.13 and 5.20. The clusters formed for host 192.168.10.101 in the Synthetic Normal Dataset forms a linear function from low duration, low packet count flows to high duration, high packet count flows. Comparatively, host 192.168.10.101 in the Synthetic Attack Dataset, does not follow a linear function, with clusters forming around low duration low packet count, low duration and high packet count, and high duration and low packet count. Given a large proportion of the implemented network attacks increase the frequency of packets in short bursts, clustering these features, and taking a comparison between the two synthetic datasets reveals a distinction between the network data when malicious commands exist in the network. As noted, with a higher variance of BACnet address lengths, and data requests in the network, there will be a higher signal-to-noise ratio in packet sizes which will require further investigation. Due to the previously discussed proprietary middleware protocol implementation, it was not possible to compare a higher variance of packet sizes given the *Write property* commands which caused the variance in the Synthetic Datasets was not accessible in the Real Normal Dataset.

5.5 Hidden Markov Models

This section details the use of Hidden Markov Models (HMMs) to identify malicious network transactions in the Synthetic Datasets generated using the simulator defined in §4.6.5.

5.5.1 Preprocessing for Hidden Markov Model

Hidden Markov Models require transition probabilities between emissions to define a model, and sequences of emissions to train and test the model. Therefore, preprocessing was required to transform the categorical data of each network packet into an emission label, generate sequences of these labels, and then define the transition probabilities between each emission. Two emission labelling schemata were developed from a number of features from each packet. First, the features source IP, destination IP, command, and command type were used to define a unique emission for each packet henceforth referred to as the *CMD* schema. Second, the *LEN* schema was defined which supplemented the features used in the *CMD* schema with the byte length feature, to evaluate the use of packet length as a discriminative feature. Each packet in the Synthetic Normal Dataset, and the Synthetic Attack Dataset was assigned two emission labels based on the two schemata.

Sequences were defined using a fixed time length window. Two time lengths were initially investigated. First, sequences were derived using a 2 second window, then a 30 second window. Due to the periodic bursts of traffic, a feature of polling and CoV reporting, the 30 second window would capture two polling sequences in each window, and was thus discarded in favour of the 2 second window size. Using a time window rather than a fixed sequence length allows for variable length sequences to be generated, which better reflects the bursts in traffic on the network. A number of low occurring sequences were generated due to multiple communication sequences overlapping relating to the same host. For example, take normal sequences (a, b, c, d) and (e, f, g), if they occur in the same time window, a new sequence such as (a, b, e, f, c, d, g) may be derived. These sequences would not necessarily be identified if deriving the sequences from a purely theoretical model, or when inserting defined malicious sequences into a normal sequence, and as such increases the robustness of the sequencing time window approach.

Sequences were constructed at a host level specificity due to the number of emission symbols generated using the schemata. For the *CMD* schema, 460 and 524 emission symbols were identified in the Synthetic Normal and Synthetic Attack Datasets respectively, while the *LEN* schema generated 460 and 597 emission symbols. The difference only existing in the Synthetic Attack dataset symbols indicates that the length of the packet can be a distinguishing feature between normal and malicious traffic for this dataset. A limitation of the HMM implementation was that only unary symbols could be passed as a sequence, and as such, 525 and 598 unique emission symbols would be required for the *CMD* and *LEN* schemas respectively. As such, a network level HMM is left for future work.

Sequences were generated for both the Synthetic Normal and Synthetic Attack datasets based on the two schemata, for each host, resulting in two sets of sequences per host which represent the same data. The Synthetic Normal sequences were used for training each HMM, while the Synthetic Attack sequences were used to test each HMM, for each respective schema.

For the full Synthetic Normal and Synthetic Attack datasets for each host, the majority of dataset pairs had between a 22.6% and 27.8% proportional split in size. One host (the target of many attacks) had a 38.1% split. Given the differences in proportion between each dataset pair, the number of samples used for each model was normalised to have a 80% training and 20% testing data split. Further, to reduce potential sampling bias for the training dataset, each full Synthetic Normal Dataset for each host was split into ten equally-sized training sets. Each training set was then used to train a HMM resulting in ten HMMs for each host. The proportionally-sampled Synthetic Attack dataset was kept constant for testing each hosts ten models.

When a HMM is faced with an emission in the testing set which was not encountered in the training set, a transition probability is not defined. Following the approach of Ariu et al. (2007), the unique emission symbols in the testing set which do not occur in the any of the training sets were changed to a wildcard symbol. The wildcard symbol is added as a transition in each model with a zero emission transition probability from every other emission symbol. This allows the model to clearly identify emission symbols which have not been encountered previously. As noted by Ariu et al. (2007), if the emission symbols are highly representative of the dataset, then it is appropriate for the wildcard emissions to be referred to as an anomaly. Given that the Synthetic Normal Dataset only contains normal network commands, while the Synthetic Attack Dataset contains both normal and attack traffic, wildcard symbols are classed as known anomalies as they are distinct from the normal commands on the network. For classification problems with a HMM, each class to be identified requires a separate model (Ariu et al., 2007). Given the binary classification pursued in this research (normal or anomalous), only a normal model is generated for each training dataset and used to classify presented sequences from the testing dataset. To classify each sequence in the testing set as normal or anomalous, the log likelihood probability of each test sequence was derived from the trained model, with a range of discriminating thresholds used.

The accuracy of HMMs can be affected by the initial hidden state transition probabilities, the number of hidden states, and the size of the dataset (Ariu et al., 2007). For the defined models, the hidden state transition probability matrix for each model was generated pseudo-randomly using the *Python numpy* random function using a seeded value for repeatability. Further, the effect of altering the hidden model size on the classification ability of the HMM was explored using models of 10, 20 and 30 hidden states. As such, 580 models were generated for the *CMD* schema, and an additional 580 models for the *LEN* schema, for the three hidden state model types, totalling 3480 models.

The HMM's were implemented using the *Python* library *pomegranate* (Schreiber, 2018). Each model was trained using the *Baum-Welch* algorithm until the emission transition probabilities converged, or 100 iterations of training was completed, whichever occurred first. As noted by Schreiber (2018), parallelisation can be used for training, however it has a detrimental effect on training times when using small numbers of states. This detrimental behaviour occurred when training the 10-state models with an increase in time relative to the number of jobs given to the queue. For the 20-state, and 30-state models, parallelisation was used with two jobs with a significant reduction in training time compared to the 10-state model. The training times for selected models are detailed in Table 5.13.

5.5.2 Defining the testing set ground truth

Only two hosts initiate both normal and malicious traffic, host 192.168.10.101 the malicious controller, and host 192.168.10.182 the malicious sensor device. Host 192.168.10.66 is the external device which generates only malicious traffic and as such does not have normal behaviour traffic defined. A network level model could identify this type of adversary with the proposed labelling schema as the new host and commands will have new defined unique emission labels. Further, previous

Table 5.13: Hidden Markov Model training and classification durations for selected hosts

IP	Hidden States	Training Time (s)	Classification Time (s)	Training Length	Sequence	Labelling Schema
192.168.10.100	10	217.264	12	8207		*CMD*
192.168.10.100	20	28.980	112	8207		*CMD*
192.168.10.100	30	66.299	263	8207		*CMD*
192.168.10.101	10	285.080	14	8255		*CMD*
192.168.10.101	20	28.915	111	8255		*CMD*
192.168.10.101	30	63.805	262	8255		*CMD*
192.168.10.160	10	208.427	18	13,714		*CMD*
192.168.10.160	20	22.915	39	13,714		*CMD*
192.168.10.160	30	39.662	90	13,714		*CMD*
192.168.10.161	10	145.948	12	8133		*CMD*
192.168.10.161	20	8.330	32	8133		*CMD*
192.168.10.161	30	21.763	71	8133		*CMD*
192.168.10.126	10	62.253	6	8021		*CMD*
192.168.10.126	20	6.738	6	8021		*CMD*
192.168.10.126	30	13.106	10	8021		*CMD*
192.168.10.182	10	66.097	5	8076		*CMD*
192.168.10.182	20	7.034	7	8076		*CMD*
192.168.10.182	30	13.609	10	8076		*CMD*
192.168.10.100	10	232.877	11	8207		*LEN*
192.168.10.100	20	29.553	116	8207		*LEN*
192.168.10.100	30	64.470	258	8207		*LEN*
192.168.10.101	10	273.899	14	8255		*LEN*
192.168.10.101	20	32.188	118	8255		*LEN*
192.168.10.101	30	56.237	261	8255		*LEN*
192.168.10.160	10	203.398	17	13,714		*LEN*
192.168.10.160	20	24.735	40	13,714		*LEN*
192.168.10.160	30	32.659	84	13,714		*LEN*
192.168.10.161	10	140.170	12	8133		*LEN*
192.168.10.161	20	10.323	32	8133		*LEN*
192.168.10.161	30	16.818	69	8133		*LEN*
192.168.10.126	10	63.022	5	8021		*LEN*
192.168.10.126	20	7.316	8	8021		*LEN*
192.168.10.126	30	8.775	11	8021		*LEN*
192.168.10.182	10	65.249	5	8076		*LEN*
192.168.10.182	20	9.984	6	8076		*LEN*
192.168.10.182	30	10.600	11	8076		*LEN*

work in the literature has focused on identifying out-of-bounds threats from new hosts, such as that by Tonejc et al. (2016) with graph analysis. As such, the analysis of the external hosts malicious sequences was not undertaken using the HMM approach as there was no defined training data, and a model defined for this host would consist of each transition probability between emissions to be zero, and thus classify all traffic as malicious; which, while accurate, is not particularly interesting. Many of the other hosts, however, contain malicious sequences generated through responding to the malicious initiators.

All hosts had a semi-supervised method applied for identifying and classifying the malicious sequences in the testing dataset for defining the ground truth of the system. As part of the implemented attack framework when each attack launched the start and end times were recorded. This provided the duration of the attack, allowing each hosts testing data-frame to be sliced based on the periods defined by the attack framework. Each slice was adjusted to start and end on an even number to fit the 2 second window sequence generation method. The sequencing approach was then applied to each slice, generating sequences which were classed as malicious. This approach was taken for every host in the network for the malicious initiator hosts, 1244, 133, and 53 sequences were defined for the malicious controller, sensor and external device respectively for both the *CMD* schema and *LEN* schema. The unique sequences generated for each malicious host were 257, 29 and 10 for the *CMD* schema, and 301, 27 and 10 for the *LEN* schema. To determine the accuracy of each labelling approach each unique malicious sequence was searched for in the corresponding training dataset which does not contain malicious traffic. Due to the length of some attacks, the duration defined by the data-frame slice allows for some normal sequences to be contained in these slices, resulting in mis-classed sequences. On average, 42.93% of the classified malicious sequences existed in the training dataset. Consequently, these sequences were re-classified as normal sequences. The remaining identified sequences were defined as the malicious sequences for each host. All other sequences in the testing datasets for each host were then classed as normal to conclude the ground truth dataset creation. This process was undertaken for each labelling schema dataset individually, as the emission labels, sequence lengths and thus malicious sequences differ. For some hosts, the change in labelling schema further differentiates the attacks sent and received by hosts. This is evident from the size differences between same class *Write property* command attacks defined in the attack framework, and the change in unique malicious sequences, but not total malicious sequences between the two schemas. As such, different ground truth datasets are defined for each labelling schema.

5.5.3 Selected models

For presenting the results of the HMMs six hosts were selected, one for each device type, in addition to the two internal malicious hosts. Of the selected hosts, 192.168.10.100, 192.168.10.101 and 192.168.10.160 have only known malicious traffic, identified as wildcard emissions. Hosts 192.168.10.126, 192.168.10.161, and 192.168.10.182 have unknown attacks defined as part of the testing set, in addition to known attacks, further referred to as mixed traffic. Note that for this research unknown attacks are defined as a legitimate-yet-malicious command, which occurs normally in the training dataset, but is also used for a malicious action in the testing dataset. An example of this may be a *Write property* command which is sent between the same source and destination pair, and for the sake of the *LEN* labelling schema also has the same packet length.

5.5.4 Evaluation of Hidden Markov Model classifiers

Depending on the purpose of the model an appropriate measure to evaluate the classification ability of the model can be selected. For this research, the interest is in being accurate at distinguishing between anomalies (True Positives), and normal traffic (True Negative), while reducing the number of incorrectly classified traffic (False Positive and False Negative). Consequently, the True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR) and False Negative Rate (FNR) measures, defined in Equation 5.3-Equation 5.6 were selected for evaluating the performance of the models. Given that anomaly detection datasets have imbalanced classes by design, certain measures are more indicative of the performance of the classification model than others. Specifically, accuracy is a poor selection measurement for binary classification as it does not take into account the sizes of each class type (Powers, 2011). Often, the F1 measure is used to account for imbalanced datasets, as it is the harmonic mean of precision and TPR (Bekkar, Djemaa & Alitouche, 2013). While the F1 measure is an improvement over the accuracy measure, it does not evaluate the classification with respect to the true negatives (Powers, 2011; Bekkar et al., 2013). Noted by Powers (2011) and Bekkar et al. (2013), the Matthews Correlation Coefficient (MCC) (Matthews, 1975) is a single performance measure which can be used for evaluating binary classifiers that is less influenced by imbalanced classes within a dataset. Further, the MCC provides an evaluation metric of correctly classifying both classes (True Positive and True Negative), unlike the F1 measurement. Hence, the MCC measure defined in Equation 5.7 was selected as the core evaluation criterion. For comparison, the Precision and Accuracy measures, defined in Equation 5.1 and Equation 5.2 are also included when presenting results of each HMM.

$$Pr = \frac{TP}{TP + FP}$$

where

$Pr =$ Precision (Equation 5.1)

$TP =$ Quantity of True Positive

$FP =$ Quantity of False Positive

$$ACC = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

where

ACC = Accuracy

TP = Quantity of True Positive (Equation 5.2)

TN = Quantity of True Negative

FP = Quantity of False Positive

FN = Quantity of False Negative

$$TPR = \frac{TP}{(TP + FN)}$$

where

TPR = True Positive Rate (Equation 5.3)

TP = Quantity of True Positive

FN = Quantity of False Negative

$$TNR = \frac{TN}{(FP + TN)}$$

where

TNR = True Negative Rate (Equation 5.4)

TN = Quantity of True Negative

FP = Quantity of False Positive

$$FPR = \frac{FP}{(FP + TN)}$$

where

FPR = False Positive Rate (Equation 5.5)

FP = Quantity of False Positive

TN = Quantity of True Negative

$$FNR = \frac{FN}{(TP + FN)}$$

where

$FNR =$ False Negative Rate

$FN =$ Quantity of False Negative

$TP =$ Quantity of True Positive

(Equation 5.6)

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

where

$MCC =$ Matthews Correlation Coefficient

$TP =$ Quantity of True Positive

$TN =$ Quantity of True Negative

$FP =$ Quantity of False Positive

$FN =$ Quantity of False Negative

(Equation 5.7)

5.5.5 Hidden Markov Model results for *CMD* schema

The average results for each model for the *CMD* schema stated hosts are outlined in Tables 5.14 - 5.19. Of note, the classification ability of the models stays constant when increasing the number of hidden states from 20 to 30 for all models. This infers that, for this dataset, it is not optimal to generate models with more than 20 hidden states given the increase in training time for no increase in classification ability. Further, the optimal quantity of hidden states for the model lies between 10 and 20 states. Seymore, Mccallum and Rosenfeld (1999), observed a plateau and then a gradual decline in accuracy for the HMM when the number of hidden states was increased. Further evaluation of the quantity of hidden states for this dataset is the interest of future work.

No defined classifier is optimal over all hosts in the *CMD* schema models. Similarly, no quantity of hidden states is optimal for every host, rather, the classification ability is dependent on the individual host data. The best result, in terms of the MCC, for host 192.168.10.100 was 0.902 from C11 for hidden state model 20 with a TNR of 0.994, see Table 5.14b. Each classifier for both the 10-state and 20-state models correctly classifies each known anomaly in the network as outlined by the TPR of 1.0 for all classifiers in Table 5.14. For hosts 192.168.10.101 and 192.168.10.160, C11 is also the optimal classifier, with a MCC of 1.0 in both the 10-state and 20-state models as outlined in Tables 5.15 and 5.16, respectively. The results from these three hosts support H_3*: Machine learning is capable of identifying one or more known attacks against BACnet/IP networks*, where the TPR is 1.0.

For Host 192.168.10.161, classifiers C9-C11 for the 10-state model in Table 5.17a, and classifiers C10-C11 for the 20-state model in Table 5.17b obtain the same MCC and TPR values of 0.851 and 0.750 respectively. Similarly, classifiers C3-C9 and classifiers C4-C10 for the 10-state and 20-state models for host 192.168.10.126, outlined in Tables 5.18a and 5.18b, generate the same MCC value of 0.802 with a TPR 1.0. Unlike models for hosts 192.168.10.161 and 192.168.10.126, the models for host 192.168.10.182 are capable of obtaining a MCC of 1.0 for classifier C10, detailed in Table 5.19a. This result supports H_4*: Machine learning is capable of identifying one or more unknown attacks against BACnet/IP networks*, as the classifier can correctly identify a number of unknown attacks in the dataset with a FPR of 0.000.

Table 5.14: Evaluation metrics for the normal controller 192.168.10.100 with *CMD* labelling schema

(a) 10 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	Accuracy	MCC
C0	1.000	0.000	1.000	0.000	0.025	0.025	0.000
C1	1.000	0.045	0.955	0.000	0.026	0.069	0.034
C2	1.000	0.058	0.942	0.000	0.026	0.081	0.039
C3	1.000	0.064	0.936	0.000	0.027	0.088	0.041
C4	1.000	0.149	0.851	0.000	0.029	0.171	0.066
C5	1.000	0.209	0.791	0.000	0.031	0.229	0.081
C6	1.000	0.268	0.732	0.000	0.034	0.286	0.095
C7	1.000	0.293	0.707	0.000	0.035	0.311	0.101
C8	1.000	0.293	0.707	0.000	0.035	0.311	0.101
C9	1.000	0.673	0.327	0.000	0.073	0.681	0.221
C10	1.000	0.703	0.297	0.000	0.080	0.711	0.236
C11	1.000	0.993	0.007	0.000	0.787	0.993	0.884

(b) 20 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	Accuracy	MCC
C0	1.000	0.000	1.000	0.000	0.025	0.025	0.000
C1	1.000	0.000	1.000	0.000	0.025	0.025	0.000
C2	1.000	0.013	0.987	0.000	0.025	0.038	0.018
C3	1.000	0.033	0.967	0.000	0.026	0.058	0.029
C4	1.000	0.071	0.929	0.000	0.027	0.094	0.044
C5	1.000	0.088	0.912	0.000	0.027	0.111	0.049
C6	1.000	0.122	0.878	0.000	0.028	0.144	0.059
C7	1.000	0.135	0.865	0.000	0.029	0.156	0.062
C8	1.000	0.135	0.865	0.000	0.029	0.156	0.062
C9	1.000	0.295	0.705	0.000	0.035	0.313	0.102
C10	1.000	0.531	0.469	0.000	0.052	0.543	0.166
C11	1.000	0.994	0.006	0.000	0.820	0.994	0.902

(c) 30 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	Accuracy	MCC
C0	1.000	0.000	1.000	0.000	0.025	0.025	0.000
C1	1.000	0.000	1.000	0.000	0.025	0.025	0.000
C2	1.000	0.013	0.987	0.000	0.025	0.038	0.018
C3	1.000	0.033	0.967	0.000	0.026	0.058	0.029
C4	1.000	0.071	0.929	0.000	0.027	0.094	0.044
C5	1.000	0.088	0.912	0.000	0.027	0.111	0.049
C6	1.000	0.122	0.878	0.000	0.028	0.144	0.059
C7	1.000	0.135	0.865	0.000	0.029	0.156	0.062
C8	1.000	0.135	0.865	0.000	0.029	0.156	0.062
C9	1.000	0.295	0.705	0.000	0.035	0.313	0.102
C10	1.000	0.531	0.469	0.000	0.052	0.543	0.166
C11	1.000	0.994	0.006	0.000	0.820	0.994	0.902

Table 5.15: Evaluation metrics for the malicious controller 192.168.10.101 with *CMD* labelling schema

(a) 10 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.242	0.242	0.000
C1	1.000	0.012	0.988	0.000	0.244	0.251	0.049
C2	1.000	0.052	0.948	0.000	0.252	0.281	0.114
C3	1.000	0.081	0.919	0.000	0.257	0.303	0.144
C4	1.000	0.169	0.831	0.000	0.277	0.370	0.216
C5	1.000	0.213	0.787	0.000	0.288	0.403	0.248
C6	1.000	0.264	0.736	0.000	0.302	0.442	0.282
C7	1.000	0.299	0.701	0.000	0.313	0.468	0.306
C8	1.000	0.299	0.701	0.000	0.313	0.468	0.306
C9	1.000	0.641	0.359	0.000	0.470	0.728	0.549
C10	1.000	0.898	0.102	0.000	0.792	0.923	0.843
C11	1.000	1.000	0.000	0.000	1.000	1.000	1.000

(b) 20 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.242	0.242	0.000
C1	1.000	0.000	1.000	0.000	0.242	0.242	0.000
C2	1.000	0.011	0.989	0.000	0.244	0.250	0.052
C3	1.000	0.034	0.966	0.000	0.248	0.267	0.091
C4	1.000	0.077	0.923	0.000	0.257	0.300	0.141
C5	1.000	0.099	0.901	0.000	0.261	0.317	0.161
C6	1.000	0.143	0.857	0.000	0.271	0.350	0.197
C7	1.000	0.159	0.841	0.000	0.275	0.363	0.209
C8	1.000	0.159	0.841	0.000	0.275	0.363	0.209
C9	1.000	0.368	0.632	0.000	0.335	0.521	0.351
C10	1.000	0.615	0.385	0.000	0.453	0.708	0.528
C11	1.000	1.000	0.000	0.000	1.000	1.000	1.000

(c) 30 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.242	0.242	0.000
C1	1.000	0.000	1.000	0.000	0.242	0.242	0.000
C2	1.000	0.011	0.989	0.000	0.244	0.250	0.052
C3	1.000	0.034	0.966	0.000	0.248	0.267	0.091
C4	1.000	0.077	0.923	0.000	0.257	0.300	0.141
C5	1.000	0.099	0.901	0.000	0.261	0.317	0.161
C6	1.000	0.143	0.857	0.000	0.271	0.350	0.197
C7	1.000	0.159	0.841	0.000	0.275	0.363	0.209
C8	1.000	0.159	0.841	0.000	0.275	0.363	0.209
C9	1.000	0.368	0.632	0.000	0.335	0.521	0.351
C10	1.000	0.615	0.385	0.000	0.453	0.708	0.528
C11	1.000	1.000	0.000	0.000	1.000	1.000	1.000

Table 5.16: Evaluation metrics for the thermostat 192.168.10.160 with *CMD* labelling schema

(a) 10 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.020	0.020	0.000
C1	1.000	0.043	0.957	0.000	0.020	0.062	0.030
C2	1.000	0.080	0.920	0.000	0.021	0.098	0.041
C3	1.000	0.163	0.837	0.000	0.023	0.179	0.062
C4	1.000	0.514	0.486	0.000	0.040	0.524	0.143
C5	1.000	0.753	0.247	0.000	0.075	0.758	0.238
C6	1.000	0.959	0.041	0.000	0.349	0.960	0.575
C7	1.000	0.980	0.020	0.000	0.500	0.980	0.700
C8	1.000	0.980	0.020	0.000	0.500	0.980	0.700
C9	1.000	0.980	0.020	0.000	0.500	0.980	0.700
C10	1.000	0.981	0.019	0.000	0.517	0.981	0.711
C11	1.000	1.000	0.000	0.000	1.000	1.000	1.000

(b) 20 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.020	0.020	0.000
C1	1.000	0.020	0.980	0.000	0.020	0.039	0.020
C2	1.000	0.120	0.880	0.000	0.022	0.137	0.052
C3	1.000	0.172	0.828	0.000	0.024	0.188	0.064
C4	1.000	0.320	0.680	0.000	0.029	0.333	0.096
C5	1.000	0.493	0.507	0.000	0.038	0.503	0.137
C6	1.000	0.650	0.350	0.000	0.054	0.657	0.187
C7	1.000	0.730	0.270	0.000	0.069	0.735	0.224
C8	1.000	0.730	0.270	0.000	0.069	0.735	0.224
C9	1.000	0.980	0.020	0.000	0.500	0.980	0.700
C10	1.000	0.980	0.020	0.000	0.500	0.980	0.700
C11	1.000	1.000	0.000	0.000	1.000	1.000	1.000

(c) 30 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.020	0.020	0.000
C1	1.000	0.020	0.980	0.000	0.020	0.039	0.020
C2	1.000	0.120	0.880	0.000	0.022	0.137	0.052
C3	1.000	0.172	0.828	0.000	0.024	0.188	0.064
C4	1.000	0.320	0.680	0.000	0.029	0.333	0.096
C5	1.000	0.493	0.507	0.000	0.038	0.503	0.137
C6	1.000	0.650	0.350	0.000	0.054	0.657	0.187
C7	1.000	0.730	0.270	0.000	0.069	0.735	0.224
C8	1.000	0.730	0.270	0.000	0.069	0.735	0.224
C9	1.000	0.980	0.020	0.000	0.500	0.980	0.700
C10	1.000	0.980	0.020	0.000	0.500	0.980	0.700
C11	1.000	1.000	0.000	0.000	1.000	1.000	1.000

Table 5.17: Evaluation metrics for the normal VAV 192.168.10.161 with *CMD* labelling schema

(a) 10 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.125	0.125	0.000
C1	1.000	0.107	0.893	0.000	0.138	0.219	0.122
C2	1.000	0.107	0.893	0.000	0.138	0.219	0.122
C3	0.750	0.290	0.710	0.250	0.127	0.345	0.028
C4	0.750	0.541	0.459	0.250	0.184	0.567	0.190
C5	0.750	0.721	0.279	0.250	0.270	0.724	0.325
C6	0.750	0.862	0.138	0.250	0.429	0.848	0.489
C7	0.750	0.962	0.038	0.250	0.775	0.936	0.722
C8	0.750	0.962	0.038	0.250	0.775	0.936	0.722
C9	0.750	1.000	0.000	0.250	1.000	0.970	0.851
C10	0.750	1.000	0.000	0.250	1.000	0.970	0.851
C11	0.750	1.000	0.000	0.250	1.000	0.970	0.851

(b) 20 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.125	0.125	0.000
C1	1.000	0.107	0.893	0.000	0.138	0.219	0.122
C2	1.000	0.107	0.893	0.000	0.138	0.219	0.122
C3	0.750	0.276	0.724	0.250	0.125	0.333	0.019
C4	0.750	0.428	0.572	0.250	0.153	0.467	0.118
C5	0.750	0.621	0.379	0.250	0.214	0.636	0.245
C6	0.750	0.655	0.345	0.250	0.231	0.667	0.271
C7	0.750	0.655	0.345	0.250	0.231	0.667	0.271
C8	0.750	0.655	0.345	0.250	0.231	0.667	0.271
C9	0.750	0.966	0.034	0.250	0.750	0.939	0.716
C10	0.750	1.000	0.000	0.250	1.000	0.970	0.851
C11	0.750	1.000	0.000	0.250	1.000	0.970	0.851

(c) 30 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.125	0.125	0.000
C1	1.000	0.107	0.893	0.000	0.138	0.219	0.122
C2	1.000	0.107	0.893	0.000	0.138	0.219	0.122
C3	0.750	0.276	0.724	0.250	0.125	0.333	0.019
C4	0.750	0.428	0.572	0.250	0.153	0.467	0.118
C5	0.750	0.621	0.379	0.250	0.214	0.636	0.245
C6	0.750	0.655	0.345	0.250	0.231	0.667	0.271
C7	0.750	0.655	0.345	0.250	0.231	0.667	0.271
C8	0.750	0.655	0.345	0.250	0.231	0.667	0.271
C9	0.750	0.966	0.034	0.250	0.750	0.939	0.716
C10	0.750	1.000	0.000	0.250	1.000	0.970	0.851
C11	0.750	1.000	0.000	0.250	1.000	0.970	0.851

Table 5.18: Evaluation metrics for the normal sensor 192.168.10.126 with *CMD* labelling schema

(a) 10 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.143	0.857	0.000	0.333	0.400	0.218
C1	1.000	0.143	0.857	0.000	0.333	0.400	0.218
C2	1.000	0.414	0.586	0.000	0.423	0.590	0.418
C3	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C4	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C5	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C6	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C7	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C8	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C9	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C10	0.667	0.875	0.125	0.333	0.667	0.818	0.542
C11	0.333	1.000	0.000	0.667	1.000	0.833	0.522

(b) 20 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.300	0.300	0.000
C1	1.000	0.143	0.857	0.000	0.333	0.400	0.218
C2	1.000	0.143	0.857	0.000	0.333	0.400	0.218
C3	1.000	0.429	0.571	0.000	0.429	0.600	0.429
C4	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C5	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C6	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C7	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C8	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C9	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C10	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C11	0.333	1.000	0.000	0.667	1.000	0.833	0.522

(c) 30 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.300	0.300	0.000
C1	1.000	0.143	0.857	0.000	0.333	0.400	0.218
C2	1.000	0.143	0.857	0.000	0.333	0.400	0.218
C3	1.000	0.429	0.571	0.000	0.429	0.600	0.429
C4	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C5	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C6	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C7	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C8	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C9	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C10	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C11	0.333	1.000	0.000	0.667	1.000	0.833	0.522

Table 5.19: Evaluation metrics for the malicious sensor 192.168.10.182 with *CMD* labelling schema

(a) 10 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.320	0.320	0.000
C1	1.000	0.059	0.941	0.000	0.333	0.360	0.140
C2	1.000	0.235	0.765	0.000	0.381	0.480	0.299
C3	1.000	0.706	0.294	0.000	0.615	0.800	0.659
C4	1.000	0.871	0.129	0.000	0.785	0.912	0.827
C5	1.000	0.894	0.106	0.000	0.818	0.928	0.855
C6	1.000	0.894	0.106	0.000	0.818	0.928	0.855
C7	1.000	0.900	0.100	0.000	0.827	0.932	0.863
C8	1.000	0.900	0.100	0.000	0.827	0.932	0.863
C9	1.000	0.912	0.088	0.000	0.844	0.940	0.877
C10	1.000	1.000	0.000	0.000	1.000	1.000	1.000
C11	0.750	1.000	0.000	0.250	1.000	0.926	0.824

(b) 20 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.320	0.320	0.000
C1	1.000	0.059	0.941	0.000	0.333	0.360	0.140
C2	1.000	0.059	0.941	0.000	0.333	0.360	0.140
C3	1.000	0.471	0.529	0.000	0.471	0.640	0.471
C4	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C5	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C6	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C7	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C8	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C9	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C10	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C11	0.750	1.000	0.000	0.250	1.000	0.926	0.824

(c) 30 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.320	0.320	0.000
C1	1.000	0.059	0.941	0.000	0.333	0.360	0.140
C2	1.000	0.059	0.941	0.000	0.333	0.360	0.140
C3	1.000	0.471	0.529	0.000	0.471	0.640	0.471
C4	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C5	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C6	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C7	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C8	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C9	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C10	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C11	0.750	1.000	0.000	0.250	1.000	0.926	0.824

5.5.6 Hidden Markov Model results for ***LEN*** schema

Identical to the *CMD* schema the 30-state models share the same results as the 20-state models for the *LEN* schema. The results for the *LEN* schema are detailed in Tables 5.20-5.25. For host 192.168.10.100, classifier C11 for the 20-state model has the highest MCC with 0.892, however the TPR is 0.8. The classifier with the highest MCC and TPR is C4 for the 10-state model, with 1.0 TPR, 0.075 MCC and 0.943 FPR. However, this FPR makes the classifier unusable for real network detection purposes, as the resulting detection system would classify the majority of network traffic incorrectly. Host 192.168.10.101 is optimal with classifier C11 for the 10-state and 20-state models, with a 0.982 and 0.985 MCC values respectively. The TPR for classifier C11 in the 10-state model is slightly higher than the classifier C11 for the 20 state model (0.986 vs 0.983). For a TPR of 1.0, classifier C9 for both the 10-state and 20-state models are optimal, however they have a 0.342 and 0.605 FPR respectively. Of note, the *LEN* schema identifies the existence of unknown attacks in hosts 192.168.10.100 and 192.168.10.101. This can be identified through comparing the results for the TPR between the two schemata, where in the *CMD* schema, for each classifier the TPR is 1.0, inferring all known attacks, while for the *LEN* schema, there is a reduction in TPR, outlined in Tables 5.20a, 5.20b, 5.21a and 5.21b. The identified variance in TPR further supports hypothesis H_4, given the ability of the 10-state and 20-state models for host 192.168.10.100 and 192.168.10.101 to detect unknown attacks. Unlike hosts 192.168.10.100 and 192.168.10.101, host 192.168.10.160, all the evaluated classifiers have a TPR of 1.0. Further, classifiers C9 through C11 for both the 10-state and 20-state models provide a 1.0 MCC value for host 192.168.10.160. As more than just classifier C11 provides a MCC of 1.0, it infers that there are no normal sequences in the dataset which have low occurrence probabilities.

Host 192.168.10.161 has three classifiers, C10 and C11 for the 10-state, and C11 for the 20-state models which obtain an MCC of 0.603. However, these classifiers all obtain a TPR of 0.4. Classifiers C1 and C2 for both the 10-state and 20-state models provide a TPR of 1.0, however the accompanying MCC is 0.138 with a FPR of 0.889. The results for hosts 192.168.10.126 and 192.168.10.182 are identical between the two schemata. This occurs due to the lack of variance in packet lengths for these hosts, inferring that for the unknown attacks in the dataset for hosts 192.168.10.126 and 192.168.10.182, the packet length is not useful for distinguishing between normal and malicious commands.

Table 5.20: Evaluation metrics for the normal controller 192.168.10.100 with *LEN* labelling schema

(a) 10 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.031	0.031	0.000
C1	1.000	0.045	0.955	0.000	0.032	0.074	0.038
C2	1.000	0.057	0.943	0.000	0.033	0.086	0.043
C3	1.000	0.064	0.936	0.000	0.033	0.093	0.046
C4	1.000	0.155	0.845	0.000	0.036	0.181	0.075
C5	0.800	0.232	0.768	0.200	0.032	0.249	0.013
C6	0.800	0.297	0.703	0.200	0.035	0.312	0.037
C7	0.800	0.326	0.674	0.200	0.036	0.340	0.046
C8	0.800	0.326	0.674	0.200	0.036	0.340	0.046
C9	0.800	0.715	0.285	0.200	0.082	0.718	0.194
C10	0.800	0.742	0.258	0.200	0.089	0.744	0.209
C11	0.800	0.999	0.001	0.200	0.980	0.993	0.882

(b) 20 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.031	0.031	0.000
C1	1.000	0.000	1.000	0.000	0.031	0.031	0.000
C2	1.000	0.013	0.987	0.000	0.031	0.043	0.020
C3	1.000	0.033	0.967	0.000	0.032	0.063	0.032
C4	1.000	0.070	0.930	0.000	0.033	0.099	0.048
C5	1.000	0.088	0.912	0.000	0.034	0.116	0.054
C6	1.000	0.127	0.873	0.000	0.035	0.154	0.067
C7	1.000	0.140	0.860	0.000	0.036	0.167	0.071
C8	1.000	0.140	0.860	0.000	0.036	0.167	0.071
C9	0.800	0.355	0.645	0.200	0.038	0.369	0.056
C10	0.800	0.620	0.380	0.200	0.063	0.626	0.148
C11	0.800	1.000	0.000	0.200	1.000	0.994	0.892

(c) 30 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.031	0.031	0.000
C1	1.000	0.000	1.000	0.000	0.031	0.031	0.000
C2	1.000	0.013	0.987	0.000	0.031	0.043	0.020
C3	1.000	0.033	0.967	0.000	0.032	0.063	0.032
C4	1.000	0.070	0.930	0.000	0.033	0.099	0.048
C5	1.000	0.088	0.912	0.000	0.034	0.116	0.054
C6	1.000	0.127	0.873	0.000	0.035	0.154	0.067
C7	1.000	0.140	0.860	0.000	0.036	0.167	0.071
C8	1.000	0.140	0.860	0.000	0.036	0.167	0.071
C9	0.800	0.355	0.645	0.200	0.038	0.369	0.056
C10	0.800	0.620	0.380	0.200	0.063	0.626	0.148
C11	0.800	1.000	0.000	0.200	1.000	0.994	0.892

Table 5.21: Evaluation metrics for the malicious controller 192.168.10.101 with *LEN* labelling schema

(a) 10 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.242	0.242	0.000
C1	1.000	0.012	0.988	0.000	0.244	0.251	0.048
C2	1.000	0.051	0.949	0.000	0.251	0.280	0.113
C3	1.000	0.086	0.914	0.000	0.259	0.307	0.149
C4	1.000	0.192	0.808	0.000	0.283	0.388	0.233
C5	1.000	0.239	0.761	0.000	0.295	0.423	0.266
C6	1.000	0.285	0.715	0.000	0.308	0.458	0.296
C7	1.000	0.333	0.667	0.000	0.324	0.494	0.328
C8	1.000	0.333	0.667	0.000	0.324	0.494	0.328
C9	1.000	0.658	0.342	0.000	0.483	0.741	0.564
C10	0.997	0.867	0.133	0.003	0.730	0.898	0.793
C11	0.986	0.996	0.004	0.014	0.987	0.993	0.982

(b) 20 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.242	0.242	0.000
C1	1.000	0.000	1.000	0.000	0.242	0.242	0.000
C2	1.000	0.011	0.989	0.000	0.244	0.250	0.051
C3	1.000	0.033	0.967	0.000	0.248	0.267	0.090
C4	1.000	0.092	0.908	0.000	0.260	0.311	0.155
C5	1.000	0.114	0.886	0.000	0.265	0.328	0.173
C6	1.000	0.151	0.849	0.000	0.273	0.357	0.203
C7	1.000	0.168	0.832	0.000	0.277	0.369	0.215
C8	1.000	0.168	0.832	0.000	0.277	0.369	0.215
C9	1.000	0.395	0.605	0.000	0.345	0.541	0.369
C10	0.983	0.631	0.369	0.017	0.458	0.716	0.526
C11	0.983	0.998	0.002	0.017	0.995	0.995	0.985

(c) 30 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.242	0.242	0.000
C1	1.000	0.000	1.000	0.000	0.242	0.242	0.000
C2	1.000	0.011	0.989	0.000	0.244	0.250	0.051
C3	1.000	0.033	0.967	0.000	0.248	0.267	0.090
C4	1.000	0.092	0.908	0.000	0.260	0.311	0.155
C5	1.000	0.114	0.886	0.000	0.265	0.328	0.173
C6	1.000	0.151	0.849	0.000	0.273	0.357	0.203
C7	1.000	0.168	0.832	0.000	0.277	0.369	0.215
C8	1.000	0.168	0.832	0.000	0.277	0.369	0.215
C9	1.000	0.395	0.605	0.000	0.345	0.541	0.369
C10	0.983	0.631	0.369	0.017	0.458	0.716	0.526
C11	0.983	0.998	0.002	0.017	0.995	0.995	0.985

Table 5.22: Evaluation metrics for the normal thermostat 192.168.10.160 with *LEN* labelling schema

(a) 10 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.020	0.020	0.000
C1	1.000	0.045	0.955	0.000	0.021	0.064	0.031
C2	1.000	0.083	0.917	0.000	0.022	0.102	0.043
C3	1.000	0.150	0.850	0.000	0.024	0.167	0.060
C4	1.000	0.516	0.484	0.000	0.041	0.526	0.146
C5	1.000	0.767	0.233	0.000	0.082	0.771	0.251
C6	1.000	0.973	0.027	0.000	0.449	0.973	0.659
C7	1.000	0.990	0.010	0.000	0.667	0.990	0.812
C8	1.000	0.990	0.010	0.000	0.667	0.990	0.812
C9	1.000	1.000	0.000	0.000	1.000	1.000	1.000
C10	1.000	1.000	0.000	0.000	1.000	1.000	1.000
C11	1.000	1.000	0.000	0.000	1.000	1.000	1.000

(b) 20 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.020	0.020	0.000
C1	1.000	0.021	0.979	0.000	0.021	0.041	0.021
C2	1.000	0.125	0.875	0.000	0.023	0.143	0.054
C3	1.000	0.169	0.831	0.000	0.024	0.186	0.064
C4	1.000	0.323	0.677	0.000	0.030	0.337	0.098
C5	1.000	0.493	0.507	0.000	0.039	0.503	0.139
C6	1.000	0.646	0.354	0.000	0.056	0.653	0.189
C7	1.000	0.750	0.250	0.000	0.077	0.755	0.240
C8	1.000	0.750	0.250	0.000	0.077	0.755	0.240
C9	1.000	1.000	0.000	0.000	1.000	1.000	1.000
C10	1.000	1.000	0.000	0.000	1.000	1.000	1.000
C11	1.000	1.000	0.000	0.000	1.000	1.000	1.000

(c) 30 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.020	0.020	0.000
C1	1.000	0.021	0.979	0.000	0.021	0.041	0.021
C2	1.000	0.125	0.875	0.000	0.023	0.143	0.054
C3	1.000	0.169	0.831	0.000	0.024	0.186	0.064
C4	1.000	0.323	0.677	0.000	0.030	0.337	0.098
C5	1.000	0.493	0.507	0.000	0.039	0.503	0.139
C6	1.000	0.646	0.354	0.000	0.056	0.653	0.189
C7	1.000	0.750	0.250	0.000	0.077	0.755	0.240
C8	1.000	0.750	0.250	0.000	0.077	0.755	0.240
C9	1.000	1.000	0.000	0.000	1.000	1.000	1.000
C10	1.000	1.000	0.000	0.000	1.000	1.000	1.000
C11	1.000	1.000	0.000	0.000	1.000	1.000	1.000

Table 5.23: Evaluation metrics for the normal VAV 192.168.10.161 with *LEN* labelling schema

(a) 10 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.156	0.156	0.000
C1	1.000	0.111	0.889	0.000	0.172	0.250	0.138
C2	1.000	0.111	0.889	0.000	0.172	0.250	0.138
C3	0.800	0.264	0.736	0.200	0.163	0.345	0.052
C4	0.600	0.541	0.459	0.400	0.184	0.550	0.100
C5	0.600	0.721	0.279	0.400	0.270	0.703	0.242
C6	0.600	0.862	0.138	0.400	0.429	0.824	0.405
C7	0.400	0.947	0.053	0.600	0.567	0.869	0.403
C8	0.400	0.947	0.053	0.600	0.567	0.869	0.403
C9	0.400	0.967	0.033	0.600	0.667	0.886	0.458
C10	0.400	1.000	0.000	0.600	1.000	0.914	0.603
C11	0.400	1.000	0.000	0.600	1.000	0.914	0.603

(b) 20 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.156	0.156	0.000
C1	1.000	0.111	0.889	0.000	0.172	0.250	0.138
C2	1.000	0.111	0.889	0.000	0.172	0.250	0.138
C3	0.600	0.310	0.690	0.400	0.130	0.353	-0.068
C4	0.600	0.428	0.572	0.400	0.153	0.453	0.020
C5	0.600	0.621	0.379	0.400	0.214	0.618	0.159
C6	0.600	0.655	0.345	0.400	0.231	0.647	0.186
C7	0.600	0.655	0.345	0.400	0.231	0.647	0.186
C8	0.600	0.655	0.345	0.400	0.231	0.647	0.186
C9	0.600	0.931	0.069	0.400	0.600	0.882	0.531
C10	0.400	0.967	0.033	0.600	0.667	0.886	0.458
C11	0.400	1.000	0.000	0.600	1.000	0.914	0.603

(c) 30 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.156	0.156	0.000
C1	1.000	0.111	0.889	0.000	0.172	0.250	0.138
C2	1.000	0.111	0.889	0.000	0.172	0.250	0.138
C3	0.600	0.310	0.690	0.400	0.130	0.353	-0.068
C4	0.600	0.428	0.572	0.400	0.153	0.453	0.020
C5	0.600	0.621	0.379	0.400	0.214	0.618	0.159
C6	0.600	0.655	0.345	0.400	0.231	0.647	0.186
C7	0.600	0.655	0.345	0.400	0.231	0.647	0.186
C8	0.600	0.655	0.345	0.400	0.231	0.647	0.186
C9	0.600	0.931	0.069	0.400	0.600	0.882	0.531
C10	0.400	0.967	0.033	0.600	0.667	0.886	0.458
C11	0.400	1.000	0.000	0.600	1.000	0.914	0.603

Table 5.24: Evaluation metrics for the normal sensor 192.168.10.126 with *LEN* labelling schema

(a) 10 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.143	0.857	0.000	0.333	0.400	0.218
C1	1.000	0.143	0.857	0.000	0.333	0.400	0.218
C2	1.000	0.414	0.586	0.000	0.423	0.590	0.418
C3	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C4	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C5	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C6	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C7	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C8	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C9	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C10	0.667	0.875	0.125	0.333	0.667	0.818	0.542
C11	0.333	1.000	0.000	0.667	1.000	0.833	0.522

(b) 20 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.300	0.300	0.000
C1	1.000	0.143	0.857	0.000	0.333	0.400	0.218
C2	1.000	0.143	0.857	0.000	0.333	0.400	0.218
C3	1.000	0.429	0.571	0.000	0.429	0.600	0.429
C4	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C5	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C6	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C7	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C8	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C9	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C10	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C11	0.333	1.000	0.000	0.667	1.000	0.833	0.522

(c) 30 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.300	0.300	0.000
C1	1.000	0.143	0.857	0.000	0.333	0.400	0.218
C2	1.000	0.143	0.857	0.000	0.333	0.400	0.218
C3	1.000	0.429	0.571	0.000	0.429	0.600	0.429
C4	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C5	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C6	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C7	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C8	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C9	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C10	1.000	0.857	0.143	0.000	0.750	0.900	0.802
C11	0.333	1.000	0.000	0.667	1.000	0.833	0.522

Table 5.25: Evaluation metrics for the malicious sensor 192.168.10.182 with *LEN* labelling schema

(a) 10 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.320	0.320	0.000
C1	1.000	0.059	0.941	0.000	0.333	0.360	0.140
C2	1.000	0.235	0.765	0.000	0.381	0.480	0.299
C3	1.000	0.706	0.294	0.000	0.615	0.800	0.659
C4	1.000	0.871	0.129	0.000	0.785	0.912	0.827
C5	1.000	0.894	0.106	0.000	0.818	0.928	0.855
C6	1.000	0.894	0.106	0.000	0.818	0.928	0.855
C7	1.000	0.900	0.100	0.000	0.827	0.932	0.863
C8	1.000	0.900	0.100	0.000	0.827	0.932	0.863
C9	1.000	0.912	0.088	0.000	0.844	0.940	0.877
C10	1.000	1.000	0.000	0.000	1.000	1.000	1.000
C11	0.750	1.000	0.000	0.250	1.000	0.926	0.824

(b) 20 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.320	0.320	0.000
C1	1.000	0.059	0.941	0.000	0.333	0.360	0.140
C2	1.000	0.059	0.941	0.000	0.333	0.360	0.140
C3	1.000	0.471	0.529	0.000	0.471	0.640	0.471
C4	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C5	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C6	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C7	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C8	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C9	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C10	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C11	0.750	1.000	0.000	0.250	1.000	0.926	0.824

(c) 30 Hidden State Model

Classifier	TPR	TNR	FPR	FNR	Pr	ACC	MCC
C0	1.000	0.000	1.000	0.000	0.320	0.320	0.000
C1	1.000	0.059	0.941	0.000	0.333	0.360	0.140
C2	1.000	0.059	0.941	0.000	0.333	0.360	0.140
C3	1.000	0.471	0.529	0.000	0.471	0.640	0.471
C4	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C5	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C6	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C7	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C8	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C9	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C10	1.000	0.941	0.059	0.000	0.889	0.960	0.915
C11	0.750	1.000	0.000	0.250	1.000	0.926	0.824

5.6 Chapter summary

This chapter presented the core results of this research, relating to SQ2, SQ3 and SQ4. Three datasets, the Real Normal Dataset, Synthetic Normal Dataset and Synthetic Attack Dataset were evaluated. First, the Real Normal Dataset was explored using a range of unsupervised methods, including the Louvain community detection algorithm, k-means and GMM clustering, and time series analysis to describe the network and evaluate features which may be appropriate for classifying BACnetwork command data. Next, the Synthetic Normal Dataset and Synthetic Attack Dataset, the outputs of the simulator designed in §4.6.5 were evaluated and compared using the same unsupervised methods. The comparison of these sets revealed differences in the network patterns of attack data, compared to normal data for known network attacks, supporting hypothesis H_2, and SQ2. Further, the defined unknown network attacks were not distinguishable with frequency measures such as time series. Penultimately, the preprocessing for a set of Hidden Markov Models for classification of BACnet/IP network attacks was defined. Finally, evaluation results for the Hidden Markov Models were presented, highlighting the results of principal interest, namely repeatable MCC and TPR values which identify the ability to detect both known and unknown BACnet/IP attacks using a range of classification thresholds. These results provide evidence to hypotheses H_3, H_4 and H_5, which in turn support SQ3 and SQ4.

Chapter 6

Discussion

This chapter identifies the relationships between results derived from experiments conducted in this research to their corresponding hypotheses and research questions. The research questions defined in Chapter 3 are re-stated, and then addressed through evaluating the results derived in Chapters 4 and 5 in the context of the existing literature. Next, the contribution to knowledge provided by this thesis is stated with the implications outlined and elaborated. Finally, the chapter concludes with a critical review of the research process identifying areas in which the study could have been improved.

6.1 Research question outcomes

The principal research question posed was explored through four defined sub-questions:

$RQ1$ *How can known and unknown attacks against BACnet/IP based Building Automation Systems be detected?*

- $SQ1$ *Are BACnet devices exposed to known threats?*
- $SQ2$ *Do known BACnet attacks have distinguishable network patterns compared to normal BACnetwork traffic?*
- $SQ3$ *Is machine learning applicable to identify known and unknown attacks against BACnet/IP networks?*
- $SQ4$ *How accurate are machine learning approaches in detecting known and unknown attacks against BACnet/IP networks and devices?*

To answer these sub-questions, a number of hypotheses were defined, with the results presented in Chapters 4 and 5 providing evidence for the hypotheses, and thus answering the sub-questions of this research. The relationships between the research sub-questions, and the derived hypotheses are detailed in Table 6.1

Table 6.1: Research sub-questions explored during the research, and related hypotheses

Sub-question	Related Hypotheses
SQ1: Are BACnet devices exposed to known threats?	H_1*:BACnet devices are exposed to known threats*
SQ2: Do known BACnet attacks have distinguishable network patterns compared to normal BACnetwork traffic?	H_2*: Known BACnet attacks have distinguishable network patterns*
SQ3: Is machine learning applicable to identify known and unknown attacks against BACnet/IP networks?	H_3*: Machine learning is capable of identifying one or more known attacks against BACnet/IP networks*
	H_4*: Machine learning is capable of identifying one or more unknown attacks against BACnet/IP networks*
SQ4: How accurate are machine learning approaches in detecting known and unknown attacks against BACnet/IP networks and devices?	H_5*: Hidden Markov Models are more accurate at detecting unknown BACnet/IP based attacks than known BACnet/IP based attacks*

6.1.1 *SQ1: Are BACnet devices exposed to known threats?*

Hypothesis H_1*: BACnet devices are exposed to known threats* posited that BACnet devices are exposed to known threats. While researchers such as Praus et al. (2016) and Gasser et al. (2017) have presented data relating to the number of directly accessible BACnet devices over the Internet, the threats posed to specific device types have not previously been explored. Further, Holmberg (2003) presented threat models for the BACnet protocol, but did not define what classes of BACnet devices can be affected by the defined threats. This research sought to evaluate the potential effect a range of known threats can have on defined device profiles. The capabilities of five prominent device profiles were examined through identifying the objects and services implemented in 368 real devices. In §4.3, it was identified that the five most common commands are normally used by 97.8% of the retrieved devices. All five of these commands have one or more known malicious uses, identified in Table 4.6.

To explore the impact of known threats to a BACnet device, a model of a controller device was defined in Figure 4.3, with the developed STRIDE matrix of known threats applied. In total, 652 threat counts were identified as existing in the controller model, with denial of service the most common threat faced. Further, the network services in use by the surveyed devices were evaluated using the defined known attacks. It was found that control devices, which issue a greater variety of network commands are exposed to more commands which can be used for malicious means. This aligns with current literature, whereby protection of controllers and workstations is promoted over protection of sensors and actuators. With current network topologies, it is more plausible for external adversaries to interact with controller devices, as many sensors and actuators are still connected using serial media. However, it is expected that future buildings incorporated into the Internet of Things and Smart City environments will have fully-native IP connectivity. This is acknowledged by the recent proposals by the BACnet working group to implement a native IP layer in the protocol to supplement the existing virtual IP layer (ASHRAE, 2018). As such, identifying known threats to the prominent device profiles was justified. Hypothesis H_1 is accepted, given the identification of known threats in models of BACnet devices, the correlation of real-world defined BACnet devices to exploitable services, and the 76,489 unique Internet exposed BACnet devices over the past three years which utilise these services.

6.1.2 *SQ2: Do known BACnet attacks have distinguishable network patterns compared to normal BACnetwork traffic?*

To evaluate *SQ2*, hypothesis *H_2: Known BACnet attacks have distinguishable network patterns* was defined with the assumption that BACnet attacks would affect network patterns enough to be identifiable. To test this hypothesis, two simulated network datasets were developed based on a defined BAS scenario described in §4.6.5. The first dataset, denoted Synthetic Normal Dataset, contained only normal network data. Comparatively, the second dataset, denoted Synthetic Attack Dataset, contained Normal, known attack and unknown attack network data. The two datasets are statistically different, with the Wilcoxon signed-rank on a comparison of command frequencies over the same period reporting a W value of 249,151.5 ($p<0.1$). Consequently, the Null hypothesis, $SyntheticNormal == SyntheticAttack$ was rejected. A range of unsupervised methods were used to evaluate the two datasets, including unsupervised clustering, graph analysis and time-series analysis.

K-means and Gaussian Mixture Model (GMM) clustering were undertaken to identify, and compare patterns in network flows constructed from the two datasets. The k-means clustering approach identified different underlying structures for each dataset. The k-means result was validated by the GMM clusters, which identified the same underlying structures. Clustering the Synthetic Normal Dataset revealed a linear-based function for cluster placements, where flows moved from low duration, low packet count clusters, to high duration, high packet count clusters, visible in Figure 5.13. Similar behaviour was not replicated in the Synthetic Attack Dataset depicted in Figure 5.20, highlighting the inherrent differences between normal and malicious BACnet/IP traffic. Further, a direct comparison between BACnet commands, and the packet size feature of these commands was undertaken. Similar feature evaluation was performed in Kaur et al. (2015) and Tonejc et al. (2016), where packet lengths were normalised as a feature for anomaly detection. The packet lengths for each command in the Synthetic Normal Dataset and Synthetic Attack Dataset packet were compared. Packet length was deemed appropriate for use as a feature for identifying a number of *Write property* class attacks in the Synthetic Attack Dataset. As noted in §5.4.1, the variance in addressing scheme will increase the signal-to-noise ratio of using packet length as a feature. However, the variations in packet lengths further supports that there are distinctions between normal and malicious BACnet/IP network traffic.

Differences between the two datasets were evaluated with directed graph analysis and a time-series analysis of network packets. The Louvain community detection algorithm revealed distinct visual characteristics between the directed graphs of the two datasets. However, an expected increase in modularity classes for the Synthetic Attack Dataset did not occur. Rather, the network attacks diluted the certainty of the community detection classifiers due to the increase in communication between nodes, in addition to the normal communication actions. Time-series analysis of the full network capture for each dataset clearly identifies frequency peaks when known network attacks occur, as detailed in Figure 5.22. Graphical anomaly detection was previously conducted in Tonejc et al. (2016), with encouraging results when identifying new malicious hosts. This research extends graphical analysis of BACnet anomalies to identify previously normal hosts which also send a range of malicious traffic. The cumulative result identified from each applied unsupervised method proves that known BACnet/IP attacks have different network patterns to normal network traffic, and thus H_2 was accepted.

6.1.3 *SQ3: Is machine learning applicable to identify known and unknown attacks against BACnet/IP networks?*

Unsupervised machine learning algorithms have previously been applied to BACnet anomaly detection by Tonejc et al. (2016) for identifying known network attacks, such as new commands, new hosts, and out of context values. Comparatively, this research explored the application of two further machine learning algorithms, Artificial Neural Networks (ANNs) and Hidden Markov Models (HMMs), to both known and unknown attacks. To provide an answer to the proposed research question, two hypotheses were derived, namely, *H_3: Machine learning is capable of identifying one or more known attacks against BACnet/IP networks* and *H_4: Machine learning is capable of identifying one or more unknown attacks against BACnet/IP networks.*

Hypothesis H_3 was accepted, based on the ability to classify known network attacks by the two explored algorithms. In §4.4.2, an ANN was proven capable of identifying one class of known network attack, supporting H_3. Further, the mean results from 10 sampled HMMs for six BACnet hosts, presented in Tables 5.14 - 5.25 identified the capability of HMMs to classify known attacks at varying levels of accuracy. These results align with those presented in Tonejc et al. (2016), stating that known attacks can be identified with unsupervised machine learning methods. Unlike other studies, this research also investigated the ability to detect unknown BACnet/IP attacks. In this context, unknown attacks are defined as a legitimate command, which occurs normally in the training dataset, but can also be used for a malicious action in the testing dataset. An example of this is a *Write property* command, which normally occurs between two hosts in the training dataset, but is also used as an attack in the testing dataset. Outlined by the True Positive

Rate values in Tables 5.17 - 5.19 and Tables 5.23 - 5.25, the implemented HMMs are capable of identifying unknown BACnet/IP attacks with a range of defined classifiers. Further, evaluating the effect of the two schemata revealed that the models for hosts 192.168.10.100 and 192.168.10.101 for the *CMD* schema also identified unknown network attacks. Classification of unknown malicious BACnet commands is novel. Unlike other BACnet/IP anomaly detection studies, such as Tonejc et al. (2016) and Esquivel-Vargas et al. (2017), which focussed on identifying deviations from learned rules, this research has classified a range of in-bounds unknown attacks correctly, with acceptable true negative, and false positive rates.

6.1.4 *SQ4: How accurate are machine learning approaches in detecting known and unknown attacks against BACnet/IP networks and devices?*

Hypothesis H_5*: Hidden Markov Models are more accurate at detecting unknown BACnet/IP based attacks than known BACnet/IP based attacks* was defined to evaluate the accuracy of detecting known and unknown BACnet/IP attacks using HMMs. Given that anomaly detection is generally used to identify yet unknown threats (Goldstein & Uchida, 2016), the ability to detect unknown attacks was selected as the test criterion for this hypothesis. As described in §5.5.1, two labelling schemata, three hidden state model types and twelve classifier thresholds were used as comparative measures for the HMM approach. The models for hosts 192.168.10.100, 192.168.10.101, 192.168.10.161, 192.168.10.126 and 192.168.10.182 evaluated both known and unknown attacks in their datasets. The dataset for host 192.168.10.160 only contained known network attacks, and thus models for host 192.168.10.160 only evaluated known network attacks.

For both schemata, all models which evaluated the unknown attacks were capable of detection given the TPR result of 1.0, outlined in Tables 5.20,5.15, 5.17, 5.18 and 5.19. The most accurate classifications for the *CMD* schema were C11 for the 10-state and 20-state models for host 192.168.10.101, with a MCC of 1.0. Comparatively, the most accurate classification for the *LEN* schema was C10 for the 10-state model for host 192.168.10.182, also with a MCC of 1.0. Host 192.168.10.160 obtained a MCC of 1.0 for C11 in the 10-state and 20-state models for the *CMD* schema. Additionally, host 192.168.10.160 obtained a MCC of 1.0 for C9, C10 and C11 for the 10-state and 20-state models for the *LEN* schema. Of the presented models, four classifiers obtained a MCC of 1.0 for only the known attacks, while three classifiers obtained a MCC of 1.0 for both the known and unknown attacks. Due to the spread of attacks in the network, no examined host evaluated only unknown attack traffic. As such, the models examined with the selected evaluation metrics reported were independently, equally accurate at detecting both known and unknown attacks in the examined dataset. These results do not support the premise of hypothesis H_5, and as such hypothesis H_5 was rejected.

Unlike the results presented in Ariu et al. (2007), increasing the quantity of hidden states in this research did not consistently increase the accuracy measurements of the model. For the results in this study, all evaluation metrics plateau between the 20-state and 30-state models, with no change recorded. Further inspecting the results in Ariu et al. (2007), identical behaviour can be identified where the 20-state and 30-state models for two examined classifiers output identical evaluation metrics. This further validates the results of the applied HMM in this research, given the same behaviour is represented in both studies.

The cumulative answers of the defined research sub-questions provide an answer to the primary research question, *RQ1: How can known and unknown attacks against BACnet/IP based Building Automation Systems be detected?*. Each method explored was capable of detecting known network attacks against BACnet/IP devices at various levels of specificity. For a high-level view of network patterns, time-series analysis of the network flows was useful for visual analysis of known network attacks when compared to a known normal baseline. Both the clustering and graph analysis approaches were useful for identifying differences in network traffic between the Synthetic Normal Dataset and the Synthetic Attack Dataset. However, both methods operating in an unsupervised mode are only suitable for exploratory analysis, given the lack of ground truth. Tonejc et al. (2016) has explored an unsupervised clustering approach for known network attacks with favourable results. Further work is required to increase the robustness of using solely a clustering or graph analysis approach for detecting unknown network attacks. Conversely, the ANN deployed was effective at identifying a known *Write property* attack, and should be explored further. The HMM approach explored in this research was successful in detecting both known and unknown network attacks in the examined dataset with acceptable evaluation measures. Specifically, the ability to detect a range of unknown attacks in all of the examined hosts containing unknown attacks is promising.

6.2 Implications of the research

6.2.1 Threat modelling of BACnet devices

The research contributed to the existing threat modelling literature regarding BACnet systems in §4.3. Previous works including Granzer et al. (2010), Fisk (2012), and Caselli et al. (2016) have relied on the threat models presented in Holmberg (2003), however this work is not capable of capturing the identified known BACnet attacks presented in the literature over the past 15 years. The research collected a range of known attacks from the literature, and classified them using a STRIDE threat matrix. To evaluate the threat model, the functions of a controller device were modelled, with the defined threat model applied to generate threat impacts. The described threat modelling approach was published in Peacock et al. (2018), and can be applied to other device profiles using a model of each device.

6.2.2 Identification and evaluation of unknown network attacks

In §4.1, two potential vulnerabilities in the protocol were identified. The first vulnerability involved the *Change of Value (CoV)* reporting function of the BACnet protocol, which describes how devices disseminate values to other devices based on defined value thresholds. The research theorised that variables which are set by the subscribing device could be exploited to cause a denial of service attack in the network, due to the inherent trust between devices in the network. The vulnerability was evaluated using the developed threat model, obtaining the equal highest threat impact from all evaluated known attacks. Further, *CoV* reporting is used in all of the evaluated device profiles, outlined in Table 4.9. The vulnerability was examined using a testbed implementation described in §4.4.4, where the theorised denial of service behaviour was observed. To evaluate the vulnerability in a larger testbed, the vulnerability was incorporated into the attack framework defined in §4.6.9. The impact of the *CoV* attack on the larger network remains unclear, as the attack was limited in scope to prevent data generation failures from occurring in the defined scenario. However, the attack was detected using the evaluated HMMs. Although the commands used to instigate this attack were also used normally in the training dataset, additional emission symbols in the sequence window were not normal, and thus caused the malicious labelled sequence referencing the *CoV* attack to be identified using the wildcard emission symbol as an anomaly. Further evaluation of the impact of this attack is required given the potential for vendor-specific limits in the number of subscriptions each device holds.

The second identified vulnerability was in the *Priority Array* process of the BACnet protocol for commandable properties identified in §4.1. Commandable properties are defined as those whose value change causes physical action. The *Priority Array* can hold up to 16 values to be written to a device, at 16 different priority levels. The BACnet 2012 standard states that undefined behaviour may occur if more than one value is written to a device using the same priority level at the same time. As such, the behaviour was modelled using the data representation of the *Priority Array* in §4.3.1. The model identified that two devices may write to a third device at the same *Priority Array* level, with only the second device's value held by the array. To explore the *Priority Array* attack using the *BACnet open stack* a simulation was developed which followed the same logic as the defined model. The results in Table 4.11 report that the *Priority Array* behaviour is replicated in the *BACnet open stack*. Given that there are no restrictions or enforcement on devices writing to priorities by default in the BACnet 2012 standard, this exists as a significant issue for BACnet devices. Further, the *Write property* command is a core function of the protocol, and exists in all of the retrieved sensor, actuator, controller and advanced controller device descriptions. The identification of this issue further reinforces the requirement for segregation and filtering of BACnet devices connected to other networks, such as the Internet.

Similar to the *CoV* attack, the *Priority Array* attack was implemented in the attack framework defined in §4.6.9. Further, the attack was successfully identified due to the emission symbol being classed as anomalous, due to the target which was automatically selected by the attack framework being a non-normal destination for the malicious controller device. It is expected that if the attack was undertaken using a host which was normally communicated with, the evaluated schema would not be appropriate for detecting the attack. Future work should examine further features, such as inter-packet timings and contextual values such as a sliding window of known normal values written to each device for identifying this class of attack.

The initial identification of both the two described network attacks were discussed in Peacock et al. (2017), and further expanded upon in Peacock et al. (2018).

6.2.3 Application of anomaly detection approaches for BACnet/IP networks

This research evaluated a range of approaches which were appropriate for identifying anomalies in BACnet/IP traffic. The literature has identified methods of detecting out-of-bounds (known) network attacks, where packet features deviate from normal. This research contributed out-of-bounds network attack detection through applying an ANN to a known network attack using time-based features. A further contribution to known network attack detection in BACnet/IP was presented through the evaluation of HMMs.

The application of HMMs for BACnet/IP anomaly detection is a novel approach for both known and unknown network attacks. Further, two methods were developed to apply the HMM approach to BACnet/IP traffic. These methods are protocol agnostic, and can be utilised for other protocols to apply a HMM. First, a method was developed to generate feature based emission symbols, which provides a unique label for each specific BACnet packet. Two feature sets were implemented using the approach, with the method robust enough to allow selection of specific packet features and generate unique labels for each provided dataset. The method provides a means for comparing feature selection sets for sequence-based models. Second, a semi-supervised labelling method was developed which could enumerate and label generated sequences of emission symbols using pre-defined ground truth data. The labelling method provided the means to evaluate the classification capability of the HMMs. The HMMs deployed in this research provided promising results for the defined simulated scenario at a host based specificity for classification of known and unknown anomalous network traffic.

Three other anomaly detection methods were explored, namely, graph-analysis, unsupervised clustering and time-series analysis. All three methods show promise, with the ability to distinguish between datasets which contain normal and malicious network commands. The existing literature has explored unsupervised clustering and time-series analysis for anomalous network classification, and graph-analysis for known-bad hosts. Further tuning mechanisms are required to distinguish between normal and unknown network attacks, and should be the focus of future studies.

The evaluation of the ANN and *Write Property* attack was presented in Johnstone et al. (2015).

6.3 Critical review of the research

The original design of the research as described in §3.5 was to use real network traffic as a comparison metric for the generated simulation scenario. However, the variability in real networks, due to vendor specific device implementation, and the flexibility of the BACnet protocol made a comparison to a generic defined simulation scenario difficult. Specifically, it was unclear until inspection of the captured network traffic that a vendor-specific middleware BACnet implementation was also running on the network. This protocol seemed to undertake many normal functions of the BACnet protocol, but provided a level of abstraction to existing network dissectors. Given that the research focus was on BACnet/IP and the proprietary nature of the discovered protocol, the proprietary traffic was not evaluated. This reduced the ability to draw comparisons between the simulated BACnet/IP implementation and the real BACnetwork traffic. Additionally, the topology of the available real network did not provide the ability to capture sensor network data. The sensor devices also did not communicate using BACnet/IP. As such, it was deemed not feasible to compare the simulation scenario with the real network capture. However, this did not invalidate the selection of simulation as a research method for generating data. In other studies, such as Tonejc et al. (2016) and Esquivel-Vargas et al. (2017), rather than using real network data to validate the design of an implemented simulator, the real and simulated datasets are used to complement each other. Thus, the research adapted to take this approach, where the analysis of the real network traffic was useful for identifying the commands used in real networks, the structure and flow between controllers, and the diurnal network patterns of the traffic. Further, the real data was used to trial and evaluate a range of unsupervised methods to compare normal and malicious network traffic.

In addition, the task of generating data was underestimated, and as such required extensive additional time and resources. To build the underlying data to drive the simulated network communications a range of algorithms were implemented based on the defined scenario. The software packages commonly used for architectural and air-conditioning design did not have the required variable resolution to generate per second data. Further, through evaluation of the existing BACnet simulators the only valid selection was the *BACnet open stack* as it was not restrictive for development and deployment. However, for the purposes of this research, the network stack required a number of features to be implemented to create the defined network scenario. These issues however, could not realistically have been identified prior to the research commencement. Regardless, the effect they had on the research was extensive.

Finally, the variance in generation of the BACnet/IP specific network attacks could be improved. The attack framework was designed to be flexible and automated, primarily to eliminate the threat of bias in generating the research data. As such, there is a lack of human interaction with the generation of network attacks. Rabadia, Valli, Ibrahim and Baig (2017) suggest that automated scripts and human interaction can be distinguished by the properties of the network attack. Thus, the network patterns generated by the attack framework captures only automated attacks without a human element. If generating a dataset using the attack framework again, the introduction of human-driven attacks would be of interest for comparison between attack behaviour, and the classification ability of the explored methods.

Chapter 7

Conclusion

7.1 Research overview

This research aimed to investigate methods for identifying known and unknown network attacks against BACnet/IP managed Building Automation Systems (BASs). The research area was defined in Chapter 1, where the exploration of further anomaly detection approaches was identified, with respect to existing approaches presented in the literature. The literature presented a number of methods for detecting known network attacks in BACnet/IP networks. Focus has been on detecting known attacks, where malicious traffic have distinct features, such as increased frequencies, illegal byte values, or are sent out of sequence, when compared to normal network behaviour. Evaluating the detection of legitimate-yet-malicious commands, those which operate normally but through the protocols design or implementation can have malicious effect, has previously not been explored. Further, limited existing research has explored machine learning methods for anomaly detection in BACnet/IP networks.

Similar to other anomaly detection research areas, datasets for evaluating anomaly detection approaches in BACnetworks are limited in scope and quantity. Given the criticality of BAS networks, existing studies evaluated datasets generated using simulation test-beds, or real datasets with synthetic attacks implemented. Often these test-beds were small in scale, implement few known attacks, or did not follow existing real-network topologies. Further, open source simulation implementations were lacking in features, and require significant development to generate meaningful datasets.

The research followed five phases. The first phase investigated the BACnet/IP protocol to build an understanding of the protocol, and identify potential network commands which could be used to cause malicious action. Further, the known attacks against BACnet were retrieved from the surrounding literature for analysis in later phases. A range of existing BACnet simulator and network generation solutions were collected for generating the required network data for the research. Further, a range of existing Internet-wide scans of BACnet/IP devices were retrieved, in addition to vendor-defined device specifications for analysis.

Phase two analysed the retrieved network scans, and identified the common services and objects used in deployed device profiles. From the retrieved known attacks and the protocol investigation undertaken in phase one, a threat model was constructed. The threat model was applied to a model of a BACnet/IP controller device, in addition to the identified services operating in the vendor-defined device profiles. Further, to evaluate the retrieved network simulators, and trial implementing known and unknown network attacks, two exploratory datasets were generated following the iterative process defined in Chapter 3. Ultimately, the *BACnet open stack* was selected due to its flexibility. A range of algorithms and methods were selected for evaluation. These were, Artificial Neural Networks (ANNs), Hidden Markov Models (HMMs), graph analysis, k-means clustering, Gaussian Mixture Model clustering and time series analysis. Both ANNs and time series analysis were conducted on the trial dataset with a defined known attack, providing encouraging results.

Phase three consisted of collecting real BACnetwork data from an Australasian university, and designing the simulation environment. A simulation scenario, underlying data generation algorithms, network topology, and attack framework were designed. The simulation was then deployed to generate and capture both normal and anomalous network traffic for evaluation. Two datasets were generated, the Synthetic Normal Dataset, which consisted of one month of normal network traffic, and the Synthetic Attack Dataset, consisting of one week of normal traffic with pseudo-randomly generated anomalous network traffic drawn from the defined attack framework.

In phase four, experiments were conducted to evaluate the graph analysis, clustering, time series and HMM approaches for detecting anomalous network traffic. Preprocessing was undertaken for each dataset to allow the application of the selected algorithms and methods.

During the final phase, the outputs generated by each algorithm-dataset pairing were evaluated. The results of this evaluation, in addition to the results generated in phase two, tested the posed hypotheses and provided answers which supported the research questions. Further, the results identified directions for future work.

7.2 Summary of contributions

7.2.1 Are BACnet devices exposed to known threats?

Existing literature has presented a range of attacks which can be used against BACnet devices. Additionally, Holmberg (2003) identified a range of threats to the BACnet protocol. Further, Praus et al. (2016) and Gasser et al. (2017) identified that BACnet devices are openly accessible over the Internet, and readily accessible through services such as *Shodan*. Existing research however, has not explored the threats faced by specific BACnet device types. This research explored the impact existing known threats from the literature could have on BACnet devices through the creation of a STRIDE threat model. Further, two additional threats were identified and evaluated in the context of the threat model. It was proven that BACnet devices are exposed to known threats. Further, BACnet controller and workstation device types face more potential known threats than lower-level sensor and actuator device types. This finding is in-line with current literature suggestions, where protection of controller and workstation devices is promoted over protection of sensor and actuator devices. It is expected that future BASs, which will be connected to the Internet of Things and Smart City environments will shift perceptions of importance in regards to device protection. Therefore, future work should aim to protect all types of devices.

7.2.2 Do known BACnet attacks have distinguishable network patterns compared to normal BACnetwork traffic?

This research identified distinguishable network patterns between generated normal and malicious BACnet commands. Unsupervised clustering of network flows identified differences in packet quantity and flow sizes between known BACnet attacks which involve changes in frequency, such as denial of service and flooding attacks. These patterns were further identified using time series analysis, where comparisons of command frequencies identified spikes when network attacks were undertaken. Graph analysis using an unsupervised community detection algorithm detailed differences in network structures between networks which have only normal commands, and those with a mixture of normal and malicious commands. However, network hosts which conduct malicious commands were not identifiable with only graph based analysis. The cumulative results from the explored approaches support the existing research paradigm, where deviation from the normal properties of BACnet packets are used to identify anomalies.

7.2.3 Is machine learning applicable to identify known and unknown attacks against BACnet/IP networks?

Machine learning previously has had limited use in the domain of BACnet/IP anomaly detection. Further, research conducted by Tonejc et al. (2016) focused solely on known network attacks. In contrast, this research explored the application of ANNs for anomaly detection for known network attacks, and HMMs for anomaly detection for both known and unknown network attacks. It was concluded that both approaches are capable of detecting known network attacks with acceptable certainty. In addition, HMMs were found to be capable of detecting unknown network attacks in the evaluated dataset. The HMM results are promising and HMMs should be evaluated with further datasets containing unknown network attacks to generalise the results of this research.

7.2.4 How accurate are machine learning approaches in detecting known and unknown attacks against BACnet/IP networks and devices?

Previous applications of machine learning were found to be highly accurate in BACnet/IP anomaly detection. The approaches explored in this research were selected due to their high accuracy rates in other anomaly detection problems. Evaluation of the accuracy of the selected approaches was undertaken using multiple metrics to further examine the suitability in classifying both normal and anomalous traffic correctly. This research identified that both ANNs and HMMs are accurate at detecting known network attacks with similar results to those presented in Tonejc et al. (2016), and can distinguish between known attacks and normal traffic. Further, the research discerns that HMMs can be equally accurate at detecting unknown attacks, in addition to distinguishing between unknown attack and normal network traffic. The HMM results presented follow the trend of those presented Ariu et al. (2007).

The cumulative answers of the explored research sub-questions provided an answer to the primary research question, *RQ1: How can known and unknown attacks against BACnet/IP based Building Automation Systems be detected?*. As outlined in Table 7.1, four of the five hypotheses were accepted. H_5 was not supported, as the HMMs explored were equally accurate at detecting both known and unknown network attacks. Rather, the HMMs explored were not more, or less, accurate at detecting known and unknown attacks in the explored dataset. Ultimately, the results of the study were positive, and contributed to the domain knowledge through the creation of a generalisable threat model for BACnet, identification and evaluation of two unknown network attacks, and the exploration of a range of anomaly detection techniques capable of detecting known and unknown BACnet/IP network attacks.

Table 7.1: Summary results of research sub-questions and related hypotheses explored during the research

Sub-question	Related Hypotheses	Result
SQ1: Are BACnet devices exposed to known threats?	H_1*:BACnet devices are exposed to known threats*	Accepted
SQ2: Do known BACnet attacks have distinguishable network patterns compared to normal BACnetwork traffic?	H_2*: Known BACnet attacks have distinguishable network patterns*	Accepted
SQ3: Is machine learning applicable to identify known and unknown attacks against BACnet/IP networ9ks?	H_3*: Machine learning is capable of identifying one or more known attacks against BACnet/IP networks*	Accepted
	H_4*: Machine learning is capable of identifying one or more unknown attacks against BACnet/IP networks*	Accepted
SQ4: How accurate are machine learning approaches in detecting known and unknown attacks against BACnet/IP networks and devices?	H_5*: Hidden Markov Models are more accurate at detecting unknown BACnet/IP based attacks than known BACnet/IP based attacks*	Rejected

7.3 Recommendations and future research directions

From the explored research it is recommended that BACnet devices have additional measures deployed to mitigate potential cyber-physical attack. From a technical point of view these measures should include network segregation of BACnet devices, source authentication and network monitoring. Cyber security considerations should be evaluated when connecting BACnet devices, and in general BASs, to the Internet for remote monitoring and cloud-based analytics. There should be some level of communication restriction through existing IP-based measures such as firewalls, or customised BACnet methods such as those presented in Kaur et al. (2015). Further, ensuring mutual trust between devices using source authentication measures will reduce the threat posed by many known BACnet attacks which rely on exploiting the high trust between BACnet devices in the guise of increased availability. Information Technology departments, network security departments and mechanical services departments should increase communications to improve their understanding of building control system networks which they manage. Deploying measures such as IP network monitoring would assist in identifying both potential malicious traffic, and network misconfigurations.

In terms of the future direction of research in BACnet/IP anomaly detection, identification and exploration of additional existing machine learning approaches to anomaly detection are of interest. To date, there are limited studies in terms of machine learning application to BACnet/IP, and there are various existing algorithms which could be explored, such as Support Vector Machines. This research presented an investigation into the application of HMMs and ANNs to the problem domain. There are many additional approaches within these two algorithms to explore, including feature selection tuning. Further application of an ANN to a diverse set of attacks, and deploying a network level HMM, which can handle a non-unary emission symbols are also of interest.

Dataset generation can be improved as currently there are limitations in regards to simulations, and attack generation. Research should focus on constructing testbeds using real BACnet/IP devices to generate network data. However, given the variation in device descriptions between vendors, using real devices is expected to be a costly exercise, and results may only be valid for specific vendor types. Implementing a shared cyber-range of BACnet/IP devices could be a means to improve research in this area where researchers can pool their implemented testbeds to increase the rigour and validity of methods in future studies.

The flexibility of the BACnet protocol has aided in the widespread adoption of the protocol as each vendor may design within the loose constructs of the protocol. It has however, made identifying anomalies in BACnet/IP a larger task. Specification-based anomaly detection approaches, such as those presented in Caselli et al. (2016) and Esquivel-Vargas et al. (2017) could be combined with methods which evaluate in-bounds network traffic, such as those presented in this research.

7.4 Further remarks

BACnet development has now identified the inherent lack of security in the protocol as an inhibiter of future BACnet use. Specifically, in June 2018 the BACnet working committee proposed Addenda *135-2016bj* for public review (ASHRAE, 2018). The aim of this review of the Addenda was increasing the native security of BACnet implementations to fit with previous iterations of the Addenda for expanding the use of native IP into the BACnet stack. The purpose of native IP is to enable all BACnet devices to operate using IP as a medium, with the aim of integration into the Internet of Things and cloud-based infrastructures. The proposal introduces the use of websockets to provide TLS to BACnet devices, specifically focusing on encryption and source authentication, allowing dynamic host address assignments. Additionally, the proposal identifies that vendors should provide the ability to update hardware firmware and software. These are positive steps that are long overdue in a slow moving but widely used protocol.

However, the proposal does not reduce the requirement for network monitoring in BACnet/IP networks. With BACnet devices set to be connected to further diverse networks and devices, detection capabilities must continue to improve to address future needs.

References

Abaid, Z., Sarkar, D., Kaafar, M. A. & Jha, S. (2016). The Early Bird Gets the Botnet: A Markov Chain Based Early Warning System for Botnet Attacks. In *Local Computer Networks (LCN), 2016 IEEE 41st Conference on* (pp. 61–68). IEEE. doi:10.1109/LCN.2016.17

Abraham, S. & Nair, S. (2015, January). Predictive Cyber Security Analytics Framework : A Non-Homogenous Markov Model for Security Quantification. *4*.

Adobe. (2018). Adobe Reader Pro 2018. Software.

Ahmad, M. W., Mourshed, M., Mundow, D., Sisinni, M. & Rezgui, Y. (2016). Building energy metering and environmental monitoring – A state-of-the-art review and directions for future research. *Energy and Buildings*, *120*, 85–102. doi:https://doi.org/10.1016/j.enbuild.2016.03.059

AIRAH. (2015a, August). Space Temperature Set Point and Control Bands. Online.

AIRAH. (2015b, August). Ventilation and Air Flow. Online.

Amazon. (2017, December). *Amazon Compute Service Level Agreement.* Retrieved from https://aws.amazon.com/ec2/sla/

Anderson, J. P. (1980). Computer security threat monitoring and surveillance. *Technical Report, James P. Anderson Company.*

Andropov, S., Guirik, A., Budko, M. & Budko, M. (2017). Network anomaly detection using artificial neural networks. In *2017 20th Conference of Open Innovations Association*. IEEE. doi:10.23919/FRUCT.2017.8071288

Antonini, A., Barenghi, A., Pelosi, G. & Zonouz, S. (2014, October). Security challenges in building automation and SCADA. In *2014 International Carnahan Conference on Security Technology (ICCST)* (pp. 1–6). doi:10.1109/CCST.2014.6986996

AnyLogic. (2017). AnyLogic. Online. Software. Retrieved from https://www.anylogic.com/downloads/

AOL. (2017). Moloch v0.19.2. Online. Software.

Applegate, S. D. (2013, June). The dawn of Kinetic Cyber. In *2013 5th International Conference on Cyber Conflict (CYCON 2013)* (pp. 1–15).

Ariu, D., Giacinto, G. & Perdisci, R. (2007). Sensing Attacks in Computers Networks with Hidden Markov Models. In P. Perner (Ed.), *Machine Learning and Data Mining in Pattern Recognition* (pp. 449–463). Berlin, Heidelberg: Springer Berlin Heidelberg.

ASHRAE. (2009). *2009 ASHRAE HANDBOOK FUNDAMENTALS* (SI) (M. S. Owen, Ed.). ASHRAE.

ASHRAE. (2012a). *2012 ASHRAE HANDBOOK HVAC Systems and Equipment* (SI) (M. S. Owen, Ed.). ASHRAE.

ASHRAE. (2012b). *ASHRAE Guideline 14: Measurement of Energy and Demand Savings*. ANSI/ASHRAE.

ASHRAE. (2018, June). Proposed Addendum bj to Standard 135-2016 BACnet - A Data Communication Protocol for Building Automation and control networks. Online. Retrieved from http://www.bacnet.org/Addenda/Add-135-2016bj-ppr2-draft-28_chair_approved.pdf

Attorney Generals Office. (2010). *Critical Infrastructure Resilience Strategy*.

Axelsson, S. (2000). *Intrusion detection systems: A survey and taxonomy*.

Baig, Z. A., Szewczyk, P., Valli, C., Rabadia, P., Hannay, P., Chernyshev, M., . . . Peacock, M. (2017, August). Future challenges for smart cities: Cyber-security and digital forensics. *Digital Investigation*, *22*, 3–13. doi:https://doi.org/10.1016/j.diin.2017.06.015

Bauer, M. (2012, December). pycryptopan 0.01. Online. Software. Retrieved from https://pypi.org/project/pycryptopan/#description

Bekkar, M., Djemaa, H. K. & Alitouche, T. A. (2013). Evaluation Measures for Models Assessment over Imbalanced Data Sets. *Journal of Information Engineering and Applications*, *3*(10), 27–37.

Benbenishti, L. (2017, April). *SCADA MODBUS Protocol Vulnerabilities*. Retrieved from https://www.cyberbit.com/scada-modbus-protocol-vulnerabilities/

Bernier, M. (2013). *Military Activities and Cyber Effects (MACE) Taxonomy*. Defence Research, Development Canada, Centre for Operational Research and Analysis.

Bhattacharyya, D. K. & Kalita, J. K. (2014). Network Anomaly Detection : A Machine Learning Perspective. Taylor and Francis.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R. & Lefebvre, E. (2008, October). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2008*(P10008), 12. doi:10.1088/1742-5468/2008/10/P10008

Bockholt, C. (2017). *Markov Chain analysis of packet sequence for intrusion detection* (Master's thesis, Iowa State University). Retrieved from https://lib.dr.iastate.edu/etd/15264/

Bowers, B. (2013). How to Own a Building: Exploiting the Physical World with BACnet and the BACnet Attack Framework. In *ShmooCon*.

BTL. (2017). *Bacnet Testing Laboratories Implementation Guidelines* (Guidelines No. 0.47). Bacnet Testing Laboratories.

BTL. (2018). *BTL Listing of Tested Products*. Retrieved from http://www.bacnetinternational.net/btl/

Byres, E. J., Franz, M. & Miller, D. (2004). The use of attack trees in assessing vulnerabilities in SCADA systems. In *Proceedings of the International Infrastructure Survivability Workshop*.

Callegari, C., Giordano, S. & Pagano, M. (2014). Neural network based anomaly detection. In *IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks*. IEEE. doi:10.1109/CAMAD.2014.7033256

Cárdenas, A. A., Amin, S. & Sastry, S. (2008). Research Challenges for the Security of Control Systems. In *Proceedings of the 3rd Conference on Hot Topics in Security* (6:1–6:6). HOTSEC'08. San Jose, CA: USENIX Association. Retrieved from http://dl.acm.org/citation.cfm?id=1496671.1496677

Caselli, M. (2015, August). *Teleconference Regarding BAS*. Private Communication.

Caselli, M. (2016, November). *Intrusion Detection in Networked Control Systems: From System Knowledge to Network Security* (Doctoral dissertation, Univ. of Twente, Enschede).

Caselli, M., Zambon, E., Amann, J., Sommer, R. & Kargl, F. (2016). Specification Mining for Intrusion Detection in Networked Control Systems. In *25th USENIX Security Symposium (USENIX Security 16)* (pp. 791–806). Austin, TX: USENIX Association. Retrieved from https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/caselli

Caselli, M., Zambon, E. & Kargl, F. (2015). Sequence-aware Intrusion Detection in Industrial Control Systems. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security* (pp. 13–24). CPSS '15. Singapore, Republic of Singapore: ACM. doi:10.1145/2732198.2732200

CBMS. (2015a). CBMS Engineering Tool. Online. Software. Retrieved from http://www.cbmsstudio.com/store/p7/Engineering_Tool.html

CBMS. (2015b). CBMS Studio V1.3.7.1204. Online. Software. Retrieved from http://www.cbmsstudio.com/bacnet-simulator.html

Čeleda, P., Krejčí, R. & Krmíček, V. (2012). Flow-Based Security Issue Detection in Building Automation and Control Networks. In R. Szabó & A. Vidács (Eds.), *Information and Communication Technologies* (Chap. 7, Vol. 7479, pp. 64–75). Lecture Notes in Computer Science. Springer Berlin Heidelberg. doi:10.1007/978-3-642-32808-4_7

Censys. (2018). Retrieved from https://censys.io/

Chandola, V., Banerjee, A. & Kumar, V. (2012, May). Anomaly Detection for Discrete Sequences: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, *24*(5), 823–839. doi:10.1109/TKDE.2010.235

Cheminod, M., Durante, L. & Valenzano, A. (2013). Review of Security Issues in Industrial Networks. *Industrial Informatics, IEEE Transactions on*, *9*(1), 277–293. doi:10.1109/tii.2012.2198666

Chipkin, P. (2009). *Bacnet for Field Technicians*. Chipkin Automation Systems.

Christensen, L. B., Johnson, B. & Turner, L. A. (2011). *Research methods, design, and analysis* (11th). Allyn & Bacon.

Cisco. (2008, August). *Building Automation System over IP (BAS/IP) Design and Implementation Guide*. Cisco, Johnson Controls. Retrieved from https://www.cisco.com/c/dam/en_us/solutions/industries/docs/trec/jControls_DIG.pdf

Cleveland, R. B., Cleveland, W. S., McRae, J. E. & Terpenning, I. (1990, å). STL: A Seasonal-Trend-Decomposition Procedure Based on Loess. *Journal of Official Statistics*, *6*(1), 3–73. Retrieved from https://www.wessa.net/download/stl.pdf

Combs, G. (2017). Dumpcap v2.2.6. Online. Software. Wireshark.

Contemporary Controls. (2014). BASview A Small but Powerful Building Management System User Manual. Online. Version 1.42d. Contemporary Controls. Retrieved from http://www.ccontrol.com/pdf/TD1105000M.pdf

Crotty, M. (1998). *The foundations of social research : meaning and perspective in the research process*. London Sage Publications.

Curnow, J. & Curnow, D. (2014, January). *What size air conditioner do I need*. Retrieved from http://www.airandwater.com.au/blog/size-air-conditioner-need/

Dainotti, A., Pescapé, A., Rossi, P. S., Palmieri, F. & Ventre, G. (2008). Internet traffic modeling by means of Hidden Markov Models. *Computer Networks*, *52*(14), 2645–2662.

Degelman, L. (2004). Advanced building simulation. In A. Malkawi & G. Augenbroe (Eds.), (Chap. 3: Simulation and uncertainty: Weather predictions). Routledge.

Denning, D. E. (1987, February). An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, *13*(2), 222–232.

Distech. (2010). *BACnet Fundamentals Course* (tech. rep. No. 801). Distech Controls. pdf course.

Dooley, K. & Dooley, K. (2001). Social Research Methods. In *4 th ed. Upper Saddle River, NJ.* Prentice Hall.

Dua, S. & Du, X. (2011). *Data Mining and Machine Learning in Cybersecurity* (1st). Boston, MA, USA: Auerbach Publications.

Dussel, P., Gehl, C., Laskov, P., Buber, J.-U., Stormann, C. & Kastner, J. (2010). Cyber-Critical Infrastructure Protection Using Real-Time Payload-Based Anomaly Detection. In E. Rome & R. Bloomfield (Eds.), *Critical Information Infrastructures Security* (Vol. 6027, pp. 85–97). Lecture Notes in Computer Science. Springer Berlin Heidelberg. doi:10.1007/978-3-642-14379-3_8

Edgar, T. & Manz, D. (2017). *Research Methods for Cyber Security*. Cambridge, MA, USA: Elsevier Science.

EnergyPlus. (2018). EnergyPlus. Online. Software. Retrieved from https://energyplus.net

Esquivel-Vargas, H., Caselli, M. & Peter, A. (2017). Automatic Deployment of Specification-based Intrusion Detection in the BACnet Protocol. In *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy* (pp. 25–36). CPS '17. Dallas, Texas, USA: ACM. doi:10.1145/3140241.3140244

Facilities.net. (2015, January). *Survey Suggests Many BAS could be Vulnerable to Hackers*. Online Survey. Retrieved from http://www.facilitiesnet.com/buildingautomation/article/Survey-Suggests-Many-BAS-Could-Be-Vulnerable-To-Hackers--15561?source=part

FBI. (2012, July). Vulnerabilities in Tridium Niagara Framework Result in Unauthorized Access to a New Jersey Company's Industrial control system. Online. Situational Information Report

Federal Bureau of Investigation. Retrieved from http://info.publicintelligence.net/FBI-AntisecICS.pdf

Fisk, D. (2012). Cyber security, building automation, and the intelligent building. *Intelligent Buildings International*, *4*(3), 169–181. doi:10.1080/17508975.2012.695277

Fovino, I. N., Coletta, A., Carcano, A. & Masera, M. (2012, October). Critical State-Based Filtering System for Securing SCADA Network Protocols. *IEEE Transactions on Industrial Electronics*, *59*(10), 3943–3950. Carcano2011 implementation paper. doi:10.1109/TIE.2011.2181132

Galliers, R. D. (1990). Choosing appropriate Information Systems Research Approaches: A Revised Taxonomy. In *IFIP TC8 WG8.2* (pp. 155–73).

Gasser, O., Scheitle, Q., Rudolph, B., Denis, C., Schricker, N. & Carle, G. (2017). The Amplification Threat Posed by Publicly Reachable BACnet Devices. *Journal of Cyber Security and Mobility*, *6*(4), 77–104. doi:10.13052/jcsm2245-1439.614

Gephi. (2017). Gephi 0.9.2. Online. Software. Retrieved from https://gephi.org

Ghosh, A. K. & Schwartzbard, A. (1999). A Study in Using Neural Networks for Anomaly and Misuse Detection. In *Proceedings of the 8th Conference on USENIX Security Symposium* (Vol. 8, p. 12). SSYM'99. Washington, D.C.: USENIX Association. Retrieved from http://dl.acm.org/citation.cfm?id=1251421.1251433

Goldstein, M. & Uchida, S. (2016, April). A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. *PLOS ONE*, *11*(4), 1–31. doi:10.1371/journal.pone.0152173

Granzer, W. & Kastner, W. (2010). Communication services for secure building automation networks. In *Industrial Electronics (ISIE), 2010 IEEE International Symposium on* (pp. 3380–3385). doi:10.1109/isie.2010.5637999

Granzer, W., Kastner, W. & Reinisch, C. (2008). Gateway-free integration of BACnet and KNX using multi-protocol devices. In *Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on* (pp. 973–978). doi:10.1109/indin.2008.4618243

Granzer, W., Praus, F. & Kastner, W. (2010). Security in Building Automation Systems. *Industrial Electronics, IEEE Transactions on*, *57*(11), 3622–3630. doi:10.1109/tie.2009.2036033

Grubb, B. (2013, May). Australia Google office building hacked. *Sydney Morning Herald*. Retrieved from http://www.smh.com.au/technology/technology-news/australian-google-office-building-hacked-20130506-2j416.html

Guba, E. G. & Lincoln, Y. S. (1994). Competing paradigms in qualitative research. In N. K. Denzin & Y. S. Lincoln (Eds.), *Handbook of Qualtiative Research* (Chap. 6, pp. 105–117). Sage.

H. Merz, C. H., T. Hansemann. (2009). *Building Automation Communication Systems with EIB/KNX, LON and BACnet*. Springer Series on Signals and Communication Technology. Springer-Verlag Berlin Heidelberg. doi:10.1007/978-3-540-88829-1

Harp, D. & Gregory-Brown, B. (2016). *SANS 2016 State of ICS Security Survey*. SANS Institute.

Haykin, S. (2009). *Neural Networks and Learning Machines* (3rd). Upper Saddle River, NJ.: Pearson.

Hersent, O., Boswarthick, D. & Elloumi, O. (2012). *The Internet of Things Key Applications and Protocols*. John Wiley & Sons, Ltd.

Herzberg, B., Bekerman, D. & Zeifman, I. (2016, October). *Breaking Down Mirai: An IoT DDoS Botnet Analysis*. Retrieved from https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html

Hettich, S. & Bay, S. (1999). KDD Cup 1999. Online Dataset. Retrieved from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Holmberg, D. G. (2003). *BACnet Wide Area Network Security Threat Assessment*. NIST.

Holmberg, D. G., Bender, J. J. & Galler, M. A. (2006). Using the BACnet Firewall Router. *ASHRAE American Society for Heating, Refrigeration and Air Conditioning Journal*, *48*(11), 10–14.

Howard, M. & Lipner, S. (2006). *The Security Development Lifecycle*. Redmond, WA, USA: Microsoft Press.

Humphries, P. (2017, October). *Phone Call, Typical BAS Topology*. Private Communication.

Hyndman, R. J. & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice* (2nd). Otexts. Retrieved from https://otexts.org/fpp2/

IBM. (2017, September). *BACnet Gateway to IBM Watson IOT Platform 1.0.12* (Solution Overview No. 54681). IBM. Retrieved from https://www-304.ibm.com/partnerworld/gsd/solutiondetails.do?solution=54681&lc=en&stateCd=P&tab=1

IEA. (2015). *Building Energy Performance Metrics*. International Energy Agency. Retrieved from http://www.iea.org/publications/freepublications/publication/building-energy-performance-metrics.html

GO-IoT. (2018). *BACnet/IoT brings BACnet to the world of IoT*. Retrieved from http://dingo-iot.io/iot-bacnet/

ISO. (2006). ISO/IEC 14543-3 Home electronic systems (HES) architecture. ISO Standard. Retrieved from https://www.iso.org/standard/43364.html

Jabado, G. (2017, October). *Discussion, BAS mechanical operations*. Private Communication.

Jain, R. & Abouzakhar, N. S. (2012). Hidden markov model based anomaly intrusion detection. In *7th International Conference for Internet Technology and Secured Transactions* (pp. 528–533). IEEE. Retrieved from https://ieeexplore.ieee.org/document/6470866/

Johnstone, M. N., Peacock, M. & den Hartog, J. (2015). Timing attack detection on BACnet via a machine learning approach. In *Proceedings of the 13th Australian Information Security Management Conference* (pp57–64). Perth, Western Australia.

Karg, S. (2015). BACnet Protocol Stack ver 0.8.2. http://sourceforge.net/projects/bacnet/.

Karg, S. (2016). BACnet Protocol Stack ver 0.8.4. http://sourceforge.net/projects/bacnet/.

Karg, S. (2017). BACnet Protocol Stack ver 0.8.5. http://sourceforge.net/projects/bacnet/.

Kastner, W., Neugschwandtner, G., Soucek, S. & Newman, H. (2005, June). Communication Systems for Building Automation and Control. *Proceedings of the IEEE*, *93*(6), 1178–1203. doi:10.1109/JPROC.2005.849726

Kaur, J., Tonejc, J., Wendzel, S. & Meier, M. (2015). Securing BACnet's Pitfalls. In H. Federrath & D. Gollmann (Eds.), *ICT Systems Security and Privacy Protection* (Vol. 455, pp. 616–629). IFIP Advances in Information and Communication Technology. Springer International Publishing. doi:10.1007/978-3-319-18467-8_41

Kepware. (2016). Bacnet/IP Driver. Online. Kepware Inc. Retrieved from https://www.kepware.com/getattachment/97b56e02-4499-498f-af7d-a71d27bb7471/bacnet-ip-manual.pdf

Khaund, K. (2015). *CyberSecurity in Smart Buildings Inaction is Not and Option Anymore* (Market Research No. 9835-19). Frost and Sullivan. Retrieved from https://www.switchautomation.com/wp-content/uploads/2015/12/Cybersecurity-in-Smart-Buildings_-Discussion-Paper.pdf

Kim, S. D. (2017). Characterization of unknown unknowns using separation principles in case study on Deepwater Horizon oil spill. *Journal of Risk Research*, *20*(1), 151–168. doi:10.1080/13669877.2014.983949. eprint: http://dx.doi.org/10.1080/13669877.2014.983949

Kirsch, J., Goose, S., Amir, Y., Wei, D. & Skare, P. (2014, January). Survivable SCADA Via Intrusion-Tolerant Replication. *Smart Grid, IEEE Transactions on*, *5*(1), 60–70. doi:10.1109/TSG.2013.2269541

Köhler, W. (2008). *Simulation of a KNX network with EIBsec protocol extensions* (Master's thesis, Vienna University of Technology).

Kostakos, V., Ferreira, D., Goncalves, J. & Hosio, S. (2016). Modelling Smartphone Usage: A Markov State Transition Model. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (pp. 486–497). UbiComp '16. Heidelberg, Germany: ACM. doi:10.1145/2971648.2971669

Kovach, R. (2016, July). Cover Story: BAS CyberSecurity. Online. Retrieved from http://www.facilitiesnet.com/buildingautomation/article/The-Power-and-Convenience-of-Networked-BAS-Come-With-Cybersecurity-Concerns-As-Well--16712?source=part

Krebs, B. (2014, February). *Target Hackers Broke in Via HVAC Company*. Retrieved from https://krebsonsecurity.com/2014/02/target-hackers-broke-in-via-hvac-company/

Krejčí, R., Čeleda, P. & Dobrovolný, J. (2012). Traffic Measurement and Analysis of Building Automation and Control Networks. In R. Sadre, J. Novotný, P. Čeleda, M. Waldburger & B. Stiller (Eds.), *Dependable Networks and Services: 6th IFIP WG 6.6 International Conference on Autonomous Infrastructure, management, and security, aims 2012, luxembourg, luxembourg, june 4-8, 2012. proceedings* (pp. 62–73). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-30633-4_9

Luft, J. & Ingham, H. (1955). The Johari window, a graphic model of interpersonal awareness. In *Proceedings of the western training laboratory in group development*. University of California, Los Angeles.

Magar, A. (2016). *State-of-the-Art in Cyber Threat Models and Methodologies*. Defence Research and Development Canada.

Makridakis, S., Wheelwright, S. C. & Hyndman, R. J. (1998). *Forecasting Methods and Applications* (Third). Wiley & Sons, Inc.

Matthews, B. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, *405*(2), 442–451. doi:https://doi.org/10.1016/0005-2795(75)90109-9

MaxMind. (2018). GeoLite2 Free Downloadable Database. Online. Retrieved from https://dev.maxmind.com/geoip/geoip2/geolite2/

Mcfarland, C., Paget, F. & Samani, R. (2015). *The Hidden Data Economy, The marketplace for stolen digital information*. McAffee. Retrieved from https://www.mcafee.com/hk/resources/reports/rp-hidden-data-economy.pdf

Mitchell, M. L. & Jolley, J. M. (2010). *Research Design Explained* (7th). Wadsworth Publishing.

Modelica. (2018). Modelica. Online. Software. Retrieved from https://www.modelica.org

Montgomery, D. C. (2013). *Design and Analysis of Experiments* (8th). John Wiley & Sons, Inc.

Mundt, T. & Wickboldt, P. (2016). Security in building automation systems - a first analysis. In *2016 International Conference On Cyber Security And Protection Of Digital Services (Cyber Security)* (pp. 1–8). doi:10.1109/CyberSecPODS.2016.7502336

Murray, G., Johnstone, M. N. & Valli, C. (2017). The Convergence of IT and OT in Critical Infrastructure. In C. Valli (Ed.), *Proceedings of the 15th Australian Information Security Management Conference* (pp. 149–155). doi:10.4225/75/5a84f7b595b4e

NCC. (2016, February). *National Construction Code*. The Australian Building Codes Board.

Newman, H. (2010). Broadcasting BACnet, *8–12*. Retrieved from http://www.bacnet.org/Bibliography/BACnet-Today-10/Newman_2010.pdf

Newman, H. (2013). *BACnet: The Global Standard for Building Automation and Control Networks*. Momentum Press. doi:10.5643/9781606502907

Novak, T. & Gerstinger, A. (2010). Safety- and Security-Critical Services in Building Automation and Control Systems. *Industrial Electronics, IEEE Transactions on*, *57*(11), 3614–3621. doi:10.1109/tie.2009.2028364

Novak, T. & Treytl, A. (2008, September). Functional safety and system security in automation systems - a life cycle model. In *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on* (pp. 311–318). doi:10.1109/ETFA.2008.4638412

Novak, T., Treytl, A. & Palensky, P. (2007). Common approach to functional safety and system security in building automation and control systems. In *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on* (pp. 1141–1148). doi:10.1109/efta.2007.4416910

O'Neill, Z., Bailey, T., Dong, B., Shashanka, M. & Luo, D. (2013). Advanced building energy management system demonstration for Department of Defense buildings. *Annals of the New York Academy of Sciences*, *1295*(1), 44–53. doi:10.1111/nyas.12188

Best Practices in HVAC Controls. (2012, June). Online.

ISO/IEC 14908 Information Technology - Control network protocol. (2012). ISO Standard. Retrieved from https://www.iso.org/standard/60203.html

Perth, Western Australia, Australia - Sunrise, Sunset, and Daylength. (2017, November). Retrieved from https://www.timeanddate.com/sun/australia/perth

Pan, Z., Hariri, S. & Al-Nashif, Y. (2014, November). Anomaly based intrusion detection for Building Automation and Control networks. In *2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA)* (pp. 72–77). doi:10.1109/AICCSA.2014.7073181

Peacock, M. & Johnstone, M. N. (2014). An analysis of security issues in building automation systems. In *Proceedings of the 12th Australian Information Security Management Conference* (pp. 100–104). Perth, Western Australia. Retrieved from http://ro.ecu.edu.au/ism/170

Peacock, M., Johnstone, M. N. & Valli, C. (2017). Security Issues with BACnet Value Handling. In O. Camp, P. Mori & S. Furnell (Eds.), *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - Volume 1: icissp,* (pp. 546–552). INSTICC. SciTePress. doi:10.5220/0006263405460552

Peacock, M., Johnstone, M. N. & Valli, C. (2018). An Exploration of Some Security Issues Within the BACnet Protocol. In P. Mori, S. Furnell & O. Camp (Eds.), *Information Systems Security and Privacy* (Vol. 867, *1*, pp. 252–272). Springer. doi:10.1007/978-3-319-93354-2

Peterson, E. (2013). Australian HVAC system design data. Online. Retrieved from http://www.uq.id.au/e.peterson/

Phillips, A. (2014, February). Comercial Buildings Baseline Study: Model - Office Buildings. Online. Dataset. Retrieved from http://www.environment.gov.au/energy/energy-efficiency/non-residential-buildings/cbbs-model-office-buildings

Ponemon Institute. (2016). *2016 Cost of Cyber Crime Study & the Risk of Business Innovation.* Ponemon Institute.

PositiveTechnologies. (2018). *ICS SECURITY 2017 in Review.* Positive Technologies.

Powers, D. M. W. (2011). Evaluation: From precision, recall and f-measure to roc., informedness, markedness and correlation. *Journal of Machine Learning Technologies*, *2*(1), 37–63.

Praus, F., Kastner, W. & Palensky, P. (2016, September 1). Secure Control Applications in Smart Homes and Buildings. *Journal of Universal Computer Science*, *22*(9), 1249–1273. Retrieved from http://www.jucs.org/jucs_22_9/secure_control_applications_in

Rabadia, P., Valli, C., Ibrahim, A. & Baig, Z. (2017). Analysis of attempted intrusions: intelligence gathered from SSH Honeypots. In C. Valli (Ed.), *Proceedings of the 15th Australian Information Security Management Conference* (Vol. 1, pp. 26–35). doi:10.4225/75/5a839e6d1d283

Rabiner, L. R. (1990). Readings in Speech Recognition. In A. Waibel & K.-F. Lee (Eds.), (Chap. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pp. 267–296). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Retrieved from http://dl.acm.org/citation.cfm?id=108235.108253

Roberts, P. (2016, November). Let's Get Cyberphysical: Internet Attack shuts off the Heat in Finland. News Article. Retrieved from https://securityledger.com/2016/11/lets-get-cyberphysical-ddos-attack-halts-heating-in-finland/

Rose, S., Spinks, N. & Canhoto, A. I. (2015). Simulation-based research [Supplement]. In *Management Research: Applying the Principles* (Chap. 6). Routledge.

Rumsfield, D. (2002, February). DoD news Briefing - Secretary Rumsfield and Gen. Myers. Online. Retrieved from http://archive.defense.gov/Transcripts/Transcript.aspx?TranscriptID=2636

Ryan, J., Lin, M.-J. & Miikkulainen, R. (1998). Intrusion detection with neural networks. In *Advances in neural information processing systems* (pp. 943–949). Retrieved from https://papers.nips.cc/paper/1459-intrusion-detection-with-neural-networks.pdf

Salamatian, K. & Vaton, S. (2001). Hidden Markov Modeling for Network Communication Channels. In *Proceedings of the 2001 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer systems* (pp. 92–101). SIGMETRICS '01. Cambridge, Massachusetts, USA: ACM. doi:10.1145/378420.378439

Scada Engine. (2009a). BACnet Device Simulator 2.0. Online. Software. Retrieved from http://www.scadaengine.com/software6.html

Scada Engine. (2009b). BACnet OPC Client User's Manual. Online. Retrieved from http://www.scadaengine.com/Documents/BACnet%20OPC%20Client%20User%20Guide.pdf

Scarfone, K. & Hoffman, P. (2009). *Guidelines on Firewalls and Firewall Policy* (Special Publication No. SP 800-41 Rev.1). NIST. Retrieved from https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-41r1.pdf

Schneider Electric. (2015). *Andover Continuum Communications Network Ports Matrix*. Schneider Electric.

Schreiber, J. (2018). Pomegranate: fast and flexible probabilistic modeling in python. *Journal of Machine Learning Research*, *18*(164), 1–6.

Seymore, K., Mccallum, A. & Rosenfeld, R. (1999). Learning Hidden Markov Model Structure for Information Extraction. In *In AAAI 99 Workshop on Machine Learning for Information Extraction* (pp. 37–42).

Shadish, W. R., Cook, T. D. & Campbell, D. T. (2002). *Experimental and quasi-experimental designs for generalized causal inference*. Boston, MA, USA: Houghton Mifflin.

Shepard, M. N. (2013). *BACnet Server for AcquiSuite*. Bacnet Testing Laboratories. Retrieved from http://www.obvius.com/sites/obvius.com/files/BACnet_Users_Guide.pdf

Shokhirev, N. (2010). *Hidden Markov Models*. Retrieved from http://www.shokhirev.com/nikolai/abc/alg/hmm/hmm.html

Siemens. (2012). BACnet Application Guide for Consulting Engineers. Online. Siemens. Retrieved from https://w3.usa.siemens.com/buildingtechnologies/us/en/consulting-engineer/engineeradvantage/Documents/BACnet%20Application%20Guide%20for%20Consulting%20Engineers%20125-1984T.pdf

Snoonian, D. (2003). Smart buildings. *Spectrum, IEEE*, *40*(8), 18–23. doi:10.1109/mspec.2003.1222043

SSPC-135. (2012). *BACnet: A Data communciation protocol for Building Automation and Control Networks*. ANSI/ASHRAE Standard 135.

SSPC-135. (2017, January). BACnet Addenda and Companion Standards. Retrieved from http://www.bacnet.org/Addenda/

Stanford III, H. W. (2011). *HVAC water chillers and cooling towers: fundamentals, application, and operation* (2nd). CRC Press.

Stouffer, K. A., Falco, J. & Kent, K. (2007). *Guide to Supervisory Control and Data Acquisition (SCADA) and Industrial Control Systems Security.* NIST.

Stouffer, K., Pillitteri, V., Lightman, S., Abrams, M. & Hahn, A. (2015, May). *NIST Special Publication 800-82: Guide to Industrial Control Systems (ICS) Security: Supervisory control and data acquisition (scada) systems, distributed control systems (dcs), and other control system configurations such as programmable logic controllers (plc).* NIST.

Takanen, A., Demott, J. D. & Miller, C. (2008). *Fuzzing for Software Security Testing and Quality Assurance.* Artech House.

The S4 Group. (2015). *What are the Best Practices for Performing a Metasys N2 Integration via the S4 Open: BACnet-N2 Router with tridium or niagara systems (kba-01022-z8n3j2).* Retrieved from https://www.thes4group.com/faqs/what-are-the-best-practices-for-performing-a-metasys-n2-integration-via-the-s4-open-bacnet-n2-router-with-tridium-or-niagara-systems-kba-01022-z8n3j2-

Thn, L. (2017, October). EWON Application User Guide: Polling Data from a BACnet/IP device. Online. Retrieved from https://websupport.ewon.biz/sites/default/files/aug-071-0-en-polling_data_from_a_bacnet_ip_device.pdf

Thomas, G. (2008, November). Introduction to BACnet Routers. *The Extension, A technical Supplement to Control Network*, *9*(6). Retrieved from https://www.ccontrols.com/pdf/Extv9n6.pdf

Tonejc, J., Güttes, S., Kobekova, A. & Kaur, J. (2016, September 1). Machine Learning Methods for Anomaly Detection in BACnet Networks. *Journal of Universal Computer Science*, *22*(9), 1203–1224. Retrieved from http://www.jucs.org/jucs_22_9/machine_learning_methods_for

Tonejc, J., Kaur, J., Karsten, A. & Wendzel, S. (2015). Visualizing BACnet data to facilitate humans in building-security decision-making. In *International Conference on Human Aspects of Information Security, Privacy, and Trust* (pp. 693–704). Springer.

Towler, J. (2015, January). Smart building controls and energy management system trends. Presentation.

Trammell, B. (2018, April). YAF: Yet Another Flow Monitor. Online. Software. Retrieved from https://tools.netsa.cert.org/yaf/libyaf/index.html

Tridium. (2017, April). Technical Document Niagara AX-3.x BACnet Guide. Online. Tridium Inc. Retrieved from https://www.novar.com/downloads/Manuals/Software/Opus/Bacnet.pdf

Trochim, W. M. (2006). *The Research Methods Knowledge Base.* Retrieved from http://www.socialresearchmethods.net/kb/

Turner, J. (2016). *A Method for Evaluating the Efficacy of Network Protocol Fuzzing Tools.*

Turner, S. & Chen, L. (2011, March). *Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms* (Informational No. 6151). Internet Engineering Task Force. Retrieved from https://tools.ietf.org/html/rfc6151

UTC Fire and Security. (2015). BACnet Protocol Guide for 2X Series Control Panels. Online. UTC Fire and Security. Retrieved from https://www.utcfssecurityproductspages.eu/apps/plugins/documentation/resources/upload/00-3243-505-0503-01-bacnet-protocol-guide-for-2x-control-panels-en-5712.pdf

Van Mieghem, P. (2009). *Performance Analysis of Communications Networks and Systems* (1st). New York, NY, USA: Cambridge University Press.

Vijayan, J. (2014, February). Retrieved from http://www.computerworld.com/article/2487452/cybercrime-hacking/target-attack-shows-danger-of-remotely-accessible-hvac-systems.html

Viking Controls. (2002). BACnet Device Object Dictionary. Online. Viking Controls. Retrieved from http://vikingcontrols.com/_documents/product/694150.PDF

Warrender, C., Forrest, S. & Pearlmutter, B. (1999). Detecting Intrusions using system calls: alternative data models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy* (pp. 133–145). IEEE. IEEE. doi:10.1109/SECPRI.1999.766910

Welch, G. & Bishop, G. (2006). *An Introduction to the Kalman Filter*. Chapel Hill, NC, USA: University of North Carolina at Chapel Hill.

White, T., Johnstone, M. N. & Peacock, M. (2017, May). An investigation into some security issues in the DDS messaging protocol. In C. Valli (Ed.), *Proceedings of the 15th Australian Information Security Management Conference* (16, pp. 132–139). doi:10.4225/75/5a84fcff95b52

Williamson, K. (2013). *Research methods : information, systems and contexts* (First edition) (K. Williamson & G. Johanson, Eds.). Prahran, VIC Tilde Publishing and distribution.

Wood, L. (2017). Global Building Management System Market 2016-2023. Online News. Cision PR Newswire. Retrieved from https://www.prnewswire.com/news-releases/global-building-management-system-market-2016-2023-300492250.html

Yu, Z. & Tsai, J. (2011). *Intrusion Detection A Machine Learning Approach*. Series in Electrical and Computer Engineering. Imperial College Press.

Zachary, J., Brooks, R. & Thompson, D. (2002). *Secure Integration of Building Networks into the Global Internet*. Pennsylvania State University.

Zhu, B., Joseph, A. & Sastry, S. (2011, October). A Taxonomy of Cyber Attacks on SCADA Systems. In *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on cyber, physical and social computing* (pp. 380–388). doi:10.1109/iThings/CPSCom.2011.34

Appendix A

BACnet Device Profiles

Table A.1: Definitions of sensor devices derived from the retrieved product listing statements

	SensorProfile
BIBBS	DS-RP-B, DS-WP-B, DM-DDB-B, DM-DOB-B, DS-RPM-B, DM-RD-B, DM-DCC-B, DM-TS-B, DS-COV-B
SERVICES	ConfirmedCOVNotification-Initiate, DeviceCommunicationControl-Execute, I-Am-Initiate, I-Have-Initiate, ReadProperty-Execute, ReadPropertyMultiple-Execute, ReinitializeDevice-Execute, SubscribeCOV-Execute, TimeSynchronization-Execute, UnconfirmedCOVNotification-Initiate, Who-Has-Execute, Who-Is-Execute, WriteProperty-Execute
OBJECTS	device, analog-input, analog-value, binary-input

Table A.2: Definitions of controller devices derived from the retrieved product listing statements

	ControllerProfile
BIBBS	DS-RP-B, DS-WP-B, DM-DDB-B, DM-DOB-B, DM-DCC-B, DS-RPM-B, DM-RD-B, DS-WPM-B, DM-TS-B, DS-COV-B, DM-DDB-A
SERVICES	ConfirmedCOVNotification-Initiate, DeviceCommunicationControl-Execute, I-Am-Execute, I-Am-Initiate, I-Have-Initiate, ReadProperty-Execute, ReadPropertyMultiple-Execute, ReinitializeDevice-Execute, SubscribeCOV-Execute, TimeSynchronization-Execute, UnconfirmedCOVNotification-Initiate, Who-Has-Execute, Who-Is-Execute, Who-Is-Initiate, WriteProperty-Execute, WritePropertyMultiple-Execute
OBJECTS	device, analog-value, analog-input, binary-input, binary-value, binary-output, analog-output, multi-state-value

Table A.3: Definitions of actuator devices derived from the retrieved product listing statements

	ActuatorProfile
BIBBS	DS-RP-B, DS-WP-B, DM-DDB-B, DM-DOB-B, DS-RPM-B, DS-COV-B, DM-TS-B, DM-UTC-B, DM-DCC-B, DM-RD-B, DS-WPM-B
SERVICES	ConfirmedCOVNotification-Initiate, DeviceCommunicationControl-Execute, I-Am-Initiate, I-Have-Initiate, ReadProperty-Execute, ReadPropertyMultiple-Execute, ReinitializeDevice-Execute, SubscribeCOV-Execute, TimeSynchronization-Execute, UTCTimeSynchronization-Execute, UnconfirmedCOVNotification-Initiate, Who-Has-Execute, Who-Is-Execute, WriteProperty-Execute, WritePropertyMultiple-Execute
OBJECTS	device, analog-input, binary-input, analog-value, binary-value, binary-output, analog-output, multi-state-value

Table A.4: Definitions of advanced controller devices derived from the retrieved product listing statements

	AdvancedControllerProfile
BIBBS	DS-RP-B, DS-WP-B, DM-DDB-B, DS-RPM-B, DM-TS-B, DM-DCC-B, DS-WPM-B, AE-N-I, AE-ACK-B, DM-DOB-B, DM-RD-B, DM-DDB-A, SCHED-I-B, DM-UTC-B, DS-COV-B, DS-WP-A, T-VMT-I, DS-RP-A, T-ATR-B, DM-LM-B, DM-BR-B, SCHED-E-B, DS-COV-A, DM-R-B, DS-RPM-A, DM-DOB-A
SERVICES	AcknowledgeAlarm-Execute, AddListElement-Execute, AtomicReadFile-Execute, AtomicWriteFile-Execute, ConfirmedCOVNotification-Execute, ConfirmedCOVNotification-Initiate, ConfirmedEventNotification-Initiate, DeviceCommunicationControl-Execute, I-Am-Execute, I-Am-Initiate, I-Have-Execute, I-Have-Initiate, ReadProperty-Execute, ReadProperty-Initiate, ReadPropertyMultiple-Execute, ReadPropertyMultiple-Initiate, ReadRange-Execute, ReinitializeDevice-Execute, RemoveListElement-Execute, SubscribeCOV-Execute, SubscribeCOV-Initiate, TimeSynchronization-Execute, UTCTimeSynchronization-Execute, UnconfirmedCOVNotification-Execute, UnconfirmedCOVNotification-Initiate, UnconfirmedEventNotification-Initiate, Who-Has-Execute, Who-Has-Initiate, Who-Is-Execute, Who-Is-Initiate, WriteProperty-Execute, WriteProperty-Initiate, WritePropertyMultiple-Execute
OBJECTS	device, analog-value, schedule, notification-class, calendar, binary-value, analog-input, binary-input, multi-state-value, binary-output, analog-output, file, trend-log, multi-state-input, program, multi-state-output, loop, event-enrollment

Table A.5: Definitions of workstation devices derived from the retrieved product listing statements

	Workstation Profile
BIBBS	DS-RP-B, DM-DDB-B, DM-DOB-B, DM-DDB-A, DS-WP-A, DS-RP-A, DS-RPM-A, DS-WPM-A, AE-ACK-A, AE-N-A, DM-MTS-A, AE-AS-A, AE-VM-A, AE-VN-A, DS-M-A, DS-V-A, SCHED-VM-A, T-V-A, DM-ADM-A, DM-TS-A, DS-COV-A, DM-UTC-A, DM-ANM-A, DS-WP-B, DS-RPM-B, DM-DCC-B, DS-WPM-B, DM-LM-B, T-ATR-A, DM-UTC-B, DM-LM-A, DM-TS-B, DM-RD-B, DM-ATS-A, AE-AVN-A, DM-DOB-A, DM-RD-A, SCHED-WS-A, DM-DCC-A, DM-BR-A, T-VMT-A, DM-OCD-B
SERVICES	AcknowledgeAlarm-Initiate, AddListElement-Execute, AddListElement-Initiate, AtomicReadFile-Initiate, AtomicWriteFile-Initiate, ConfirmedCOVNotification-Execute, ConfirmedEventNotification-Execute, CreateObject-Execute, CreateObject-Initiate, DEPRECATED, DeleteObject-Execute, DeviceCommunicationControl-Execute, DeviceCommunicationControl-Initiate, GetAlarmSummary-Initiate, GetEnrollmentSummary-Initiate, GetEventInformation-Initiate, I-Am-Execute, I-Am-Initiate, I-Have-Execute, I-Have-Initiate, ReadProperty-Execute, ReadProperty-Initiate, ReadPropertyMultiple-Execute, ReadPropertyMultiple-Initiate, ReadRange-Initiate, ReinitializeDevice-Execute, ReinitializeDevice-Initiate, RemoveListElement-Execute, RemoveListElement-Initiate, SubscribeCOV-Initiate, TimeSynchronization-Execute, TimeSynchronization-Initiate, UTCTimeSynchronization-Execute, UTCTimeSynchronization-Initiate, UnconfirmedCOVNotification-Execute, UnconfirmedEventNotification-Execute, Who-Has-Execute, Who-Has-Initiate, Who-Is-Execute, Who-Is-Initiate, WriteProperty-Execute, WriteProperty-Initiate, WritePropertyMultiple-Execute, WritePropertyMultiple-Initiate
OBJECTS	device, file, analog-value, binary-value, schedule

Appendix B

BIBB to Service Matching Tables

Table B.1: BIBB to services matching, adapted from SSPC-135 (2012)

BIBB	Services
DataSharing	
DS-RP-A	ReadProperty (Initiate)
DS-RP-B	ReadProperty (Execute)
DS-RPM-A	ReadPropertyMultiple (Initiate)
DS-RPM-B	ReadPropertyMultiple (Execute),
DS-WP-A	WriteProperty (Initiate),
DS-WP-B	WriteProperty (Execute),
DS-WPM-A	WritePropertyMultiple (Initiate),
DS-WPM-B	WritePropertyMultiple (Execute),
DS-COV-A	SubscribeCOV (Initiate), ConfirmedCOVNotification (Execute), Unconfirmed-COVNotification (Execute),
DS-COV-B	SubscribeCOV (Execute), ConfirmedCOVNotification (Initiate), Unconfirmed-COVNotification (Initiate),
DS-COVP-A	SubscribeCOVproperty (Execute), ConfirmedCOVNotification (Execute), Un-confirmedCOVNotification (Execute),
DS-COVP-B	SubscribeCOVProperty (Initiate), ConfirmedCOVNotification (Initiate), Un-confirmedCOVNotification (Initiate),
DS-COVU-A	ConfirmedCOVNotification (Execute),
DS-COVU-B	ConfirmedCOVNotification (Initiate),
DS-V-A	ReadProperty (Initiate),
DS-AV-A	ReadProperty (Initiate),
DS-M-A	WriteProperty (Initiate),
DS-AM-A	WriteProperty (Initiate),
DS-WG-A	WriteGroup (Initiate),
DS-WG-I-B	WriteGroup (Execute),
DS-WG-E-B	WriteGroup (Execute), WriteProperty (Initiate)
AlarmSharing	

BIBB	Services
AE-N-A	ConfirmedEventNotification (Execute),UnconfirmedEventNotification (Execute),
AE-N-I-B	ConfirmedEventNotification (Initiate),UnconfirmedEventNotification (Initiate),
AE-N-E-B	ReadProperty (Initiate),ConfirmedEventNotification (Initiate),UnconfirmedEventNotification (Initiate), AcknowledgeAlarm (Execute), GetEventInformation (Initiate), GetEventInformation (Execute)
AE-ACK-A	AcknowledgeAlarm (Initiate),
AE-ACK-B	AcknowledgeAlarm (Execute),
AE-ASUM-A	GetAlarmSummary (Initiate),
AE-ASUM-B	GetAlarmSummary (Execute),
AE-ESUM-A	GetEnrollmentSummary (Initiate),
AE-ESUM-B	GetEnrollmentSummary (Execute),
AE-INFO-A	GetEventInformation (Initiate),
AE-INFO-B	GetEventInformation (Execute),
AE-LS-A	LifeSafetyOperation (Initiate), ConfirmedEventNotification (Execute),UnconfirmedEventNotification (Execute), AcknowledgeAlarm (Initiate),
AE-LS-B	LifeSafetyOperation (Execute), ConfirmedEventNotification (Initiate), UnconfirmedEventNotification (Initiate), AcknowledgeAlarm (Execute), GetEventInformation (Execute)
AE-VN-A	ConfirmedEventNotification (Execute),UnconfirmedEventNotification (Execute),
AE-AVN-A	ConfirmedEventNotification (Execute),UnconfirmedEventNotification (Execute),
AE-VM-A	ReadProperty (Initiate), WriteProperty (Initiate)
AE-AVM-A	ReadProperty (Initiate), WriteProperty (Initiate), CreateObject (Initiate), DeleteObject (Initiate),
AE-AS-A	GetAlarmSummary (Initiate), GetEnrollmentSummary (Initiate), GetEventInformation (Initiate),
AE-ELV-A	ReadRange (Initiate),
AE-ELVM-A	ReadRange (Initiate),ReadProperty (Initiate), WriteProperty (Initiate),
AE-EL-I-B	ReadRange (Execute),
AE-EL-E-B	ConfirmedEventNotification (Execute), UnconfirmedEventNotification (Execute), ReadRange (Execute),
AE-NF-B A	AddListElement (Initiate), RemoveListElement (Initiate), ConfirmedCOVNotification (Initiate), UnconfirmedCOVNotification (Initiate), ConfirmedCOVNotification (Execute), UnconfirmedCOVNotification (Execute)

BIBB	Services
AE-NF-I-B	ConfirmedCOVNotification (Initiate) , UnconfirmedCOVNotification (Initiate), AddListElement (Execute), RemoveListElement (Execute)
Schedule	
SCHED-A	ReadProperty (Initiate),WriteProperty (Initiate), DEPRECATED,
SCHED-I-B	ReadProperty (Execute), WriteProperty (Execute), TimeSynchronization (Execute), UTCTimeSynchronization (Execute),
SCHED-E-B	ReadProperty (Execute), WriteProperty (Execute), TimeSynchronization (Execute),UTCTimeSynchronization (Execute), WriteProperty (Initiate),
SCHED-R-B	ReadProperty (Execute), TimeSynchronization (Execute),UTCTimeSynchronization (Execute),
SCHED-AVM-A	CreateObject (Initiate),DeleteObject (Initiate),ReadProperty (Initiate), WriteProperty (Initiate),
SCHED-VM-A	ReadProperty (Initiate), WriteProperty (Initiate),
SCHED-WS-A	ReadProperty (Initiate), WriteProperty (Initiate),
SCHED-WS-I-B	ReadProperty (Execute), WriteProperty (Execute), TimeSynchronization (Execute) ,UTCTimeSynchronization (Execute)
Trending	
T-VMT-A	ReadRange (Initiate), DEPRECATED,
T-VMT-I-B	ReadRange (Execute),
T-VMT-E-B	ReadRange (Execute), ReadProperty (Initiate),
T-ATR-A	ConfirmedEventNotification (Execute), UnconfirmedEventNotification (Execute), ReadRange (Initiate),
T-ATR-B	ConfirmedEventNotification (Initiate) , UnconfirmedEventNotification (Initiate), ReadRange (Execute),
T-VMMV-A	ReadRange (Initiate), DEPRECATED,
T-VMMV-I-B	ReadRange (Execute),
T-VMMV-E-B	ReadRange (Execute), ReadPropertyMultiple (Initiate),
T-AMVR-A	ConfirmedEventNotification (Initiate) , UnconfirmedEventNotification (Initiate), ReadRange (Execute),
T-AMVR-B	ConfirmedEventNotification (Initiate) , UnconfirmedEventNotification (Initiate), ReadRange (Execute),
T-V-A	ReadRange (Initiate),
T-AVM-A	CreateObject (Initiate), DeleteObject (Initiate), ReadProperty (Initiate), ReadRange (Initiate), WriteProperty (Initiate),
T-A-A	ConfirmedEventNotification (Execute), UnconfirmedEventNotification (Execute), ReadRange (Initiate), ReadRange (Execute)
DeviceManagement	
DM-DDB-A	Who-Is (Initiate), I-Am (Execute),
DM-DDB-B	Who-Is (Execute), I-Am (Initiate),

BIBB	Services
DM-DOB-A	Who-Has (Initiate), I-Have (Execute),
DM-DOB-B	Who-Has (Execute), I-Have (Initiate),
DM-DCC-A	DeviceCommunicationControl (Initiate),
DM-DCC-B	DeviceCommunicationControl (Execute),
DM-TM-A	ConfirmedTextMessage (Initiate),UnconfirmedTextMessage (Initiate),
DM-TM-B	ConfirmedTextMessage (Execute), UnconfirmedTextMessage (Execute),
DM-TS-A	TimeSynchronization (Initiate),
DM-TS-B	TimeSynchronization (Execute),
DM-UTC-A	UTCTimeSynchronization (Initiate),
DM-UTC-B	UTCTimeSynchronization (Execute),
DM-RD-A	ReinitializeDevice (Initiate),
DM-RD-B	ReinitializeDevice (Execute),
DM-BR-A	ReinitializeDevice (Initiate), CreateObject (Initiate), AtomicReadFile (Initiate), AtomicWriteFile (Initiate),
DM-BR-B	ReinitializeDevice (Execute), AtomicReadFile (Execute), AtomicWriteFile (Execute),
DM-R-A	UnconfirmedCOVNotification (Execute),
DM-R-B	UnconfirmedCOVNotification (Initiate),
DM-LM-A	AddListElement (Initiate), RemoveListElement (Initiate),
DM-LM-B	AddListElement (Execute), RemoveListElement (Execute),
DM-OCD-A	CreateObject (Initiate), DeleteObject (Initiate),
DM-OCD-B	CreateObject (Execute),DeleteObject (Execute),
DM-VT-A	VT-Open (Initiate), VT-Close (Initiate), VT-Data (Initiate), VT-Close (Execute), VT-Data (Execute)
DM-VT-B	VT-Close (Initiate), VT-Data (Initiate), VT-Open (Execute), VT-Close (Execute), VT-Data (Execute),
DM-ANM-A	Who-Is (Initiate), I-Am (Execute),
DM-ADM-A	ReadProperty (Initiate),
DM-ATS-A	TimeSynchronization (Initiate), UTCTimeSynchronization (Initiate),
DM-MTS-A	TimeSynchronization (Initiate), UTCTimeSynchronization (Initiate),
NetworkManagement	
NM-CE-A	Establish-Connection-To-Network (Initiate), Disconnect-Connection-To-Network (Initiate),
NM-CE-B	Establish-Connection-To-Network (Execute), Disconnect-Connection-To-Network (Execute),
NM-RC-A	Who-Is-Router-To-Network (Initiate), Initialize-Routing-Table (Initiate), I-Am-Router-To-Network (Execute), I-Could-Be-Router-To-Network (Execute), Initialize-Routing-Table-Ack (Execute),
NM-RC-B	Who-Is-Router-To-Network (Initiate), I-Am-Router-To-Network (Initiate), Initialize-Routing-Table-Ack (Initiate), Who-Is-Router-To-Network (Execute), I-Am-Router-To-Network (Execute), Initialize-Routing-Table (Execute)
NetworkSecurity	

BIBB	Services
NS-SD	Security-Payload (Initiate), Security-Response (Initiate), Request-Key-Update (Initiate), What-Is-Network-Number (Initiate), Network-Number-Is (Initiate), Challenge-Request (Execute), Security-Payload (Execute), Security-Response (Execute), Update-Key-Set (Execute), Update-Distribution-Key (Execute), What-Is-Network-Number (Execute), Network-Number-Is (Execute),
NS-ED	Security-Payload (Initiate), Security-Response (Initiate), Request-Key-Update (Initiate), What-Is-Network-Number (Initiate), Network-Number-Is (Initiate), Challenge-Request (Execute), Security-Payload (Execute), Security-Response (Execute), Update-Key-Set (Execute), Update-Distribution-Key (Execute), What-Is-Network-Number (Execute), Network-Number-Is (Execute),
NS-MAD	Security-Payload (Initiate), Security-Response (Initiate), Request-Key-Update (Initiate), What-Is-Network-Number (Initiate), Network-Number-Is (Initiate), Challenge-Request (Execute), Security-Payload (Execute), Security-Response (Execute), Update-Key-Set (Execute), Update-Distribution-Key (Execute), What-Is-Network-Number (Execute), Network-Number-Is (Execute),
NS-DMK-A	Request-Master-Key (Execute), Set-Master-Key (Initiate),
NS-DMK-B	Request-Master-Key (Initiate), Set-Master-Key (Execute),
NS-KS	Request-Key-Update (Execute), Request-Master-Key (Execute), Update-Key-Set (Initiate), Update-Distribution-Key (Initiate), Set-Master-Key (Initiate), Security-Payload (Initiate), Security-Response (Initiate), Request-Key-Update (Initiate), What-Is-Network-Number (Initiate),Network-Number-Is (Initiate), Challenge-Request (Execute), Security-Payload (Execute),Security-Response (Execute), Update-Key-Set (Execute), Update-Distribution-Key (Execute), What-Is-Network-Number (Execute), Network-Number-Is (Execute),
NS-TKS	Request-Key-Update (Execute), Request-Master-Key (Execute), Update-Key-Set (Initiate), Update-Distribution-Key (Initiate), Set-Master-Key (Initiate), Security-Payload (Initiate), Security-Response (Initiate), Request-Key-Update (Initiate), What-Is-Network-Number (Initiate),Network-Number-Is (Initiate), Challenge-Request (Execute), Security-Payload (Execute),Security-Response (Execute), Update-Key-Set (Execute), Update-Distribution-Key (Execute), What-Is-Network-Number (Execute), Network-Number-Is (Execute),
NS-SR	Security-Payload (Initiate), Security-Response (Initiate), Request-Key-Update (Initiate), What-Is-Network-Number (Initiate), Network-Number-Is (Initiate), Challenge-Request (Execute), Security-Payload (Execute), Security-Response (Execute), Update-Key-Set (Execute), Update-Distribution-Key (Execute), What-Is-Network-Number (Execute), Network-Number-Is (Execute),
NS-SP	Security-Payload (Initiate), Security-Response (Initiate), Request-Key-Update (Initiate), What-Is-Network-Number (Initiate), Network-Number-Is (Initiate), Challenge-Request (Execute), Security-Payload (Execute), Security-Response (Execute), Update-Key-Set (Execute), Update-Distribution-Key (Execute), What-Is-Network-Number (Execute), Network-Number-Is (Execute),