Automated discovery of BACnet relations from bus traffic

Melcher Stikkelorum

Master Thesis Cyber Security
January 2020

Supervisors
Dr.-Ing. E. Tews
Dr. A. Peter

University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands

Table of Contents

1. Introduction

A building management system (BMS) is a type of control system used to manage a building's systems and services, such as heating, ventilation and air conditioning (HVAC). Nowadays, it is common practice to include, among others, lighting, sun blind control and various security facilities into the automation system [1].

A BMS integrates all the functionality of building equipment into one centralized system and provides building operators with the tools to manage a building's energy efficiency by regular measuring and monitoring of all of the building equipment [2]. Whereas BMSs used to be analog-based, they were first computerized in the 1950s when analog systems (e.g. electrical, pneumatic or hydraulic controllers) were monitored by computers for the first time [3]. After the introduction of Programmable Logic Controllers (PLCs) in 1971, BMSs increasingly became computer controlled.

A modern BMS consists of an intricate mix of various components such as controllers, sensors and actuators, each of which performs a part of the automation. In general there are three layers that can be distinguished in a traditional BMS. The layers are shown in Figure 1. Starting at the bottom there are the sensors and actuators. These are devices that are in direct contact with the environment and/or the end-user. As such this layer is called the field layer. On the next layer up any input from the field layer is gathered and used to make automation decisions. This automation layers contains all processing and infrastructural hardware such as controllers, gateways and routers. The top layer provides management functionality. Management of a BMS can also often be done remotely over the internet.

It is apparent that components in the BMS need to communicate status and control information to other components to enable them to make automation decisions. Communication happens over a network or "bus". To illustrate this, the reading from a temperature sensor in the field layer will, for example, be used by a controller in the automation layer to decide that a heat valve must be opened or closed.

Components on different levels of the automation system often have different requirements about latency and bandwidth. To cut on cost, it was common practice for different layers to use a bus which met the minimum network requirements. Different buses would typically also require their own communication protocol. Although a large variety of communications protocols exists for BMSs, in this thesis the focus is on the BACnet (Building Automation and Control network) protocol, which is one of the most widespread building automation and control network protocols and currently holds a market share of about 60%.

Because of the different requirements at each level of the BMS, the usage of IP networks in combination with Ethernet, was regarded as unnecessary and costly. Nowadays, the widespread adoption of IP-based technologies has caused modern office buildings to be completed with an Ethernet IP backbone built-in. This allows BMSs to readily plug into the existing infrastructure and take advantage of the "free wiring" [4].

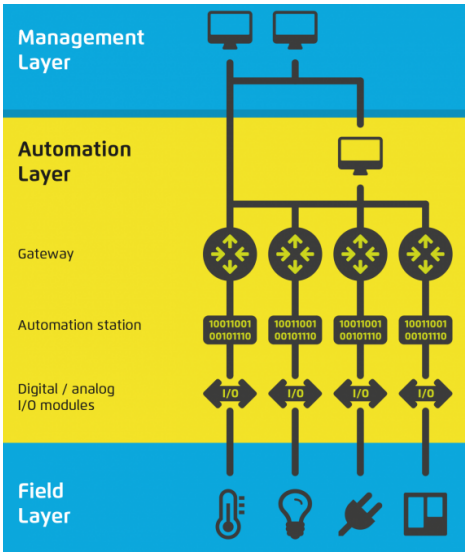

Figure 1: Various layers involved within a building automation system.

The move to IP-based networks for BMSs was supported by the introduction of BACnet/IP and allows BACnet to run over an IP network alongside other IP based technologies [5]. This ability of BMSs to communicate over IP has thus resulted in their connection to the corporate intranet which often also provides connections to the internet to allow the BMS to be operated remotely [6].

BACnet itself was designed to run on an isolated network and thus no direct attention was paid to security issues that may surface in such an environment. Only in 2008 a security extension was added to the BACnet standard called BACnet Security Services (BSS). According to the chairman of the BACnet no building automation device vendors had yet implemented the security extension into any commercial product as late as 2013 [7]. BSS is due to be replaced by BACnet Secure Connect which integrates current IT industry security standards into BACnet by the use of TLS [8].

With the limited implementation of security measures, BACnet has a large attack surface and a BACnet BMS is susceptible to attacks like snooping, spoofing and denial of service attacks [9]. Without any security having been implemented in many BACnet BMSs, an attacker can easily tap into the building intranet and monitor and/or alter the information sent by automation devices.

In contrast to a full blown disruptive attack on a BMS (e.g. DoS, flooding), which may severely hamper the functioning of said BMS, the goal of a targeted attack could be to gain undetected access to a target network for extended periods of time. Its focus is on high impact with minimal interference of the system.

It is argued that such a targeted attack can only successfully be performed by an advanced persistent threat (APT). Presumably, such an attack is very resource intensive and is executed over an extended amount of time. APTs are often associated with government actors, which are able to invest large amounts of resources to make a long term attack possible.

A targeted attack on a BMS is one where the threat agent's ambition is to infiltrate the BMS and influence automation in a building. Such an attack may, for example, aim to change a room's temperature to a certain limit, but not more than needed, to cause an adverse effect in the environment such as productivity loss or damage in raw materials. For the threat agent to succeed, a high-level of coordination and extended knowledge of the BMS under attack are required. Therefore, an important phase of a targeted attack is intelligence gathering [10].

2. BACnet

First conceived in 1987, BACnet is an open and free to use communication protocol for – like its name suggests – building automation and control networks. It enables automation devices from different BMS vendors to interface with each other in a streamlined and standardized way. The BACnet stack was designed to support a large variety of physical mediums and data link technologies (e.g. Ethernet and LonTalk). In 1995 the BACnet standard was adopted by ASHRAE and in 2001 it was also standardized in ISO 16484-5.

In short, BACnet standardized the way of sending messages in BMSs with the help of a common network and application layer (see Figure 2). Both layers will be discussed in more detail below.

<table>
<tr><td colspan="5" rowspan="3">BACnet Application Layer (APDU)</td><td>Application (7)</td></tr>
<tr><td>Presentation (6)</td></tr>
<tr><td>Session (5)</td></tr>
<tr><td colspan="5" rowspan="2">BACnet Network Layer (NPDU)</td><td>Transport (4)</td></tr>
<tr><td>Network (3)</td></tr>
<tr><td colspan="2" rowspan="2">ISO 8802-02
Type 1</td><td>IP</td><td rowspan="2">MSTP</td><td rowspan="2">PTP</td><td rowspan="2">Data-Link (2)</td></tr>
<tr><td>UDP/IP</td></tr>
<tr><td>ARCNET</td><td>ISO 8802-3
(Ethernet)</td><td>IP Supporting
Data-Link</td><td>EIA 485</td><td>EIA 232
(RS232)</td><td>Physical (1)</td></tr>
</table>

Figure 2: BACnet network layers and the OSI model. Emphasized layers in blue are part of BACnet.

2.1 Network layer

In the past, proprietary bus solutions were common in building automation networks, but the preference is shifting to an IP-based bus. Modern buildings are often built with integrated data wiring in the form of Ethernet backbones with IP capable network equipment. As such the BMS can conveniently plug in into the backbone and use it as a bus running along IP-based technologies [11].

To support the seamless move of the BACnet to an IP-based environments BACnet/IP (B/IP) was introduced in Annex J of the BACnet standard in 2004. B/IP consists of BVLL (BACnet Virtual Link Layer) as well as UDP/IP at the data link. By default UDP port 0xBAC0 (47808) is used for all communication. The BACnet network and application data units, NPDU and APDU respectively, are encapsulated within the UDP datagram as a BVLL which indicates the function of the packet [12]. BACnet devices are addressed through IP and Ethernet MAC addresses like other UDP/IP devices.

Other data links such as Master-Slave/Token-Passing (MS/TP) can still function within a B/IP network by the use of a BACnet MS/TP to B/IP router. Since devices using other data link do not have an IP address of their own (i.e. routed MS/TP traffic is identified by the IP of the router), each device is identified by a unique BACnet address [12].

2.2 Application layer

In BACnet BMS devices are modeled in an object-oriented manner (see Figure 3). Part of the object-oriented structure which BACnet uses are: Objects, Properties and Services. Each BACnet device is represented using a collection of objects each of which has a collection of attributes called properties. Through these objects the internal state of a given device can be observed over the network. Each device features at least the “Device” object. The device object contains information about the device’s operational status as well as vendor and firmware information.

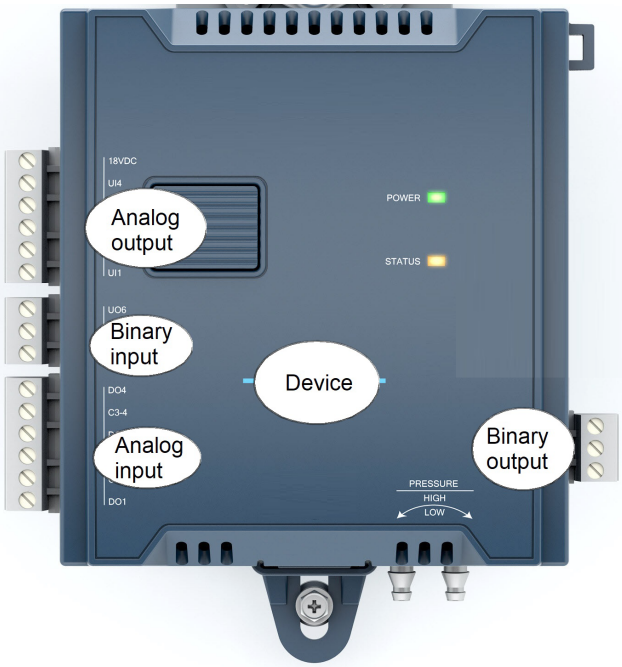

Figure 3: An example of a device modeled using BACnet objects.

A collection of common high-level object types has been standardized as part of BACnet. Each object type is identified by a number (see Table 1 for a list of examples). Vendors can implement additional proprietary object types to support additional device capabilities or functionality.

Table 1: A selection of common BACnet object names and their identifier.

Object name	**Id**	**Object name**	**Id**	**Object name**	**Id**
Analog input	0	Binary value	5	NotificationClass	15
Analog output	1	Calendar	6	Schedule	17
Analog value	2	Device	8	TrendLog	20
Binary input	3	EventEnrollment	9	EventLog	25
Binary output	4	Loop	12	Timer	31

Each BACnet device (e.g. controller) maintains a set of objects for the functionality that it can provide. These objects are always instances of a BACnet object type. Each object instance for a device is uniquely identified by its instance number. Device objects are an exception to this rule as their instance number must be unique across the entire local network. Device instance numbers are often chosen strategically such that they convey extra meaning like the building, floor and/or room that they are installed at [13].

While the names of most objects in Table 1 are self-explanatory, a few will be discussed here in more detail. The EventEnrollment object represents an event and defines the criteria for an event. The object references an object property for which an event can be generated. The EventEnrollment can be used in combination with a NotificationClass. Once an event is triggered, the NotificationClass will send event

notifications to its recipients. The event can be sent to one or more objects or broadcasted. The TrendLog object reads the present value of a monitored object periodically. Each time the present value is read, it is saved into a buffer which can be queried by other objects.

2.3 Services

In BACnet, services are a means for a controller to request information from another controller. Two groups of services can be distinguished: those that need confirmation and those that do not. A confirmed service always requires an acknowledgment to be sent in return.

Services can provide, among others, the following types functionality: alarm and events, object access, and device management. An example of an event service is the Change Of Value (COV) service through which a client can subscribe to an object's value and receive a notification upon change (SubscribeCOV and COVNotification). Examples of services providing object access are the ReadProperty and WriteProperty services which perform self-explanatory functions. These services also have a 'Multiple' variant which allows reading and writing of multiple object properties at once, respectively.

2.4 BACnet device programming and specification

The aim of BACnet is to provide a common and standardized protocol for BMS devices to communicate. Something not included in BACnet is the definition of a control language. The lack of a control language means that proprietary solutions still reign in the area of control logic creation. In this thesis, BACnet devices by the Dutch vendor Priva are analyzed. Priva supplies installation firms with a software application called 'TC Select' to allow their devices to be programmed. An installation firm is the party that configures and installs the devices as part of a complete BMS. TC Select does not directly show any BACnet constructs, but instead uses generic proprietary components and elements from which a BMS can be constructed. In this way, TC Select can also be used to program non-BACnet devices.

Components and elements are placed in graphically represented connection schemas via a drag-and-drop editor as shown in Figure 4. Once a BMS project is finished, the project is converted into firmware images and flashed to the devices.

TC Select uses a hierarchical structure to manage the objects in the BMS. It logically starts at the Building level and progresses down through building section, controller, subsystem, module. Each module's components and elements are laid out in a graphical schema. This exact hierarchy can also be seen in the left pane in Figure 4 (Verlichting groep 1 is the selected Module).

For the purpose of engineering a BMS where solutions of multiple vendors interact, the Engineering Data Exchange (EDE) file template was created. EDE is not widely adopted as it is not part of the ASHRAE BACnet standard, but, instead, was introduced by BACnet Interest Group Europe [14]. An EDE file gives engineers quick insight in what BACnet objects are available and what functionality and property values are

used in a BMS.

Files adhering to the EDE templates include a set of spreadsheet files, representing a view of BACnet object instances and their respective properties [15]. The standard attributes present in an EDE file can also be extracted from a live BMS. In the view of engineering it might sometimes be convenient or necessary to have an offline copy of this data in the form of an EDE file. Some proprietary control software solutions allow EDE files to be exported from the project. Similarly, BACnet 'explorers' allow the creation of EDE files for explored BACnet devices on the BMS. A BACnet explorer is a software application that runs on a computer connected to the BMS and can actively scan the network for BACnet devices and their objects. EDE files exported from a proprietary software tool may sometimes contain additional, non-standard, EDE attributes.

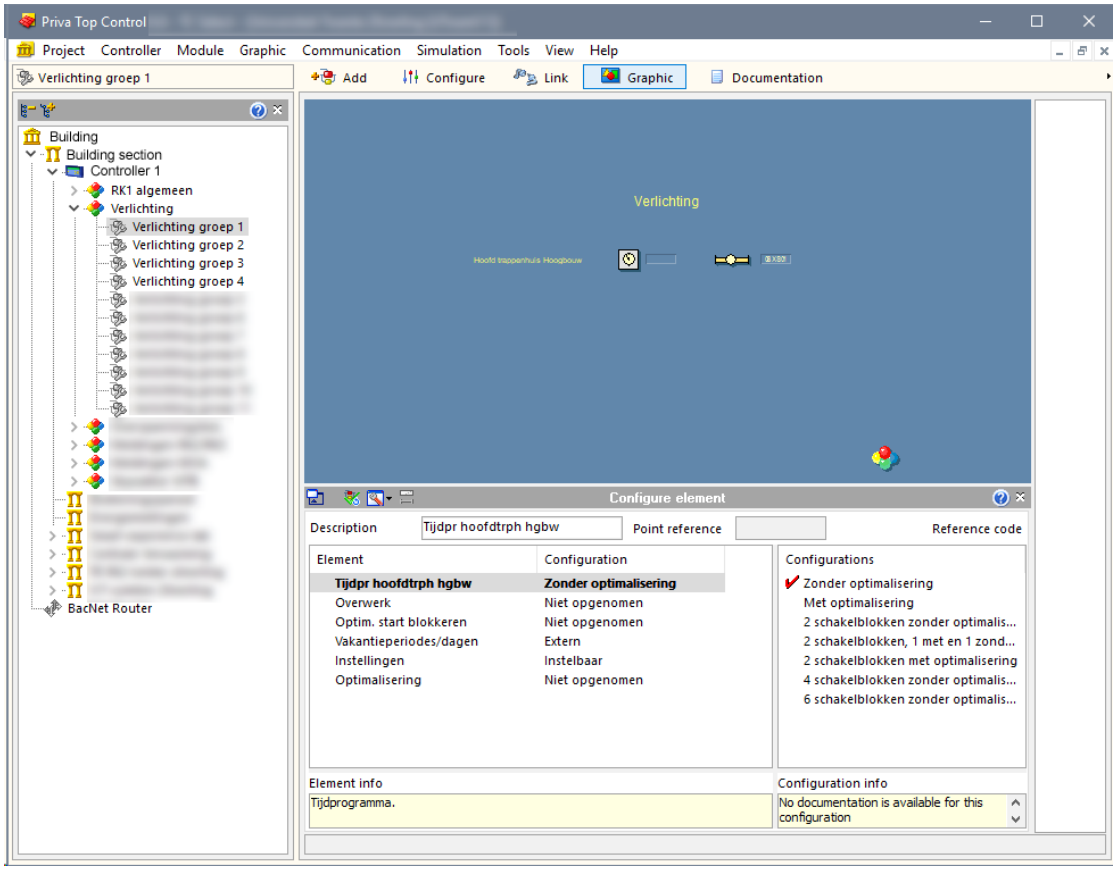

Figure 4: TC Select interface showing the graphical schema editor (middle), the project hierarchy (left), and all elements (bottom).

3. Definitions

3.1 Software modules

As discussed above, each controller in a BMS needs to be programmed. For this purpose specialized software tools, supplied by the controller's vendor, are used. The specific options of project organization in each tool differs greatly from vendor to vendor. Oftentimes, at least some modularization of the software project is possible by subdividing parts of the program. Once a software project is finalized, it can be uploaded to the controller(s).

The term **software module** comes from the software engineering domain. A monolithic application is impractical to maintain and difficult to understand. Therefore, programmers apply modularization techniques to split an application into logical chunks of functionality. Yourdon and Constantine [16] give the following definition: "A module is a lexically contiguous sequence of program statements, bounded by boundary elements, having an aggregate identifier." Boundary elements are, for example, curly braces in languages like C# and Java. This definition captures functions and classes as software modules. It is thus clear that software modules can be defined at varying granularities.

The above definition of a software module is stated very broadly. In the BMS domain different software tools allow for drastically different partitioning of a software project. Moreover, some of these applications abstract away the coding aspect of the BMS entirely. As an example, in CodeSys, a development environment for PLCs, the user must write each new function or program in a new file, whereas Priva's TC Select allows the creation of complicated modularization in the project. For this research it is assumed that two or more software modules at the same level in the hierarchy are mutually exclusive. That is, these software modules do not share any of their (lower-level) sub-modules.

3.2 Allocation of objects

In a BACnet powered software project, object instances are created to enable functionality and interoperability, i.e. if a REAL value is used in the program, a BACnet Analog Value object instance is instantiated to store this value in. This object can then be manipulated both locally and via the BACnet bus, if so desired. Depending on the software tool, the instantiation of object instances can happen in two different ways.

Explicitly: the programmer defines new instances and refers to these in his code.

Implicitly: the programmer uses high-level programming constructs – which are not BACnet specific – and the software tool translates these to object instances when necessary.

3.3 Object identifiers

The BACnet Object Identifier tuple *(T, i)* is the unique identifier for an object of type T with instance number i. An instance number is a 22-bit number. This number is often automatically allocated by the software tool starting at 0, but in some cases can be assigned manually. It must be noted that object identifiers are only unique in the scope of a given device. A special case are Device object identifiers "(8, *i*)" as these must be unique in the scope of the entire BMS with a view to message routing.

3.4 Object relationships

Using the definition of a software module above, the definition of what it means for BACnet objects to be related can be constructed. An **object relationship** is defined to exist among the objects which are part of the same software module. The objects in a software module work together to achieve the functionality of said module. Since the definition of software modules allows for different levels of granularity, so does the definition of object relationships. On a coarse level (e.g. the software project), naturally all objects are related. Moving down the ladder of granularity, objects will be subdivided into groups of related functionality and finally, at the most granular level, by subroutine.

4. Environment description

During this thesis bus traffic captures originating from a switch within a building on university campus are used. Along with bus traffic captures, the software project describing the controller programming is used. The software project describes the configuration and programming of the controllers and as such these are the main focus of this thesis. The remainder of this section will further discuss the network environment as perceived through the given traffic captures as well as the software project.

4.1 Network model

The traffic captures were created by placing a tap in the building's core communications switch. By analyzing the bus traffic captures, a network model is established as shown in Figure 5. The controllers described by the software project are encircled by a red line in the Figure. In total there are five controllers in the software project with instance numbers: 1, 2, 3, 4, 5. All traffic coming from or going to these controllers is routed through a single IP-address, 10.0.0.1, the Priva Router. The router is configured to operate in "BACnet visualization" mode; it routes all BACnet/IP traffic for the five Priva Comforte HX controllers behind it and makes them visible to the rest of the network. The router routes controllers 1, 2, 3, 4 and 5 on BACnet virtual network number 1000. Communication between the router and controllers does not have to occur strictly using BACnet/IP, but could also use different protocols.

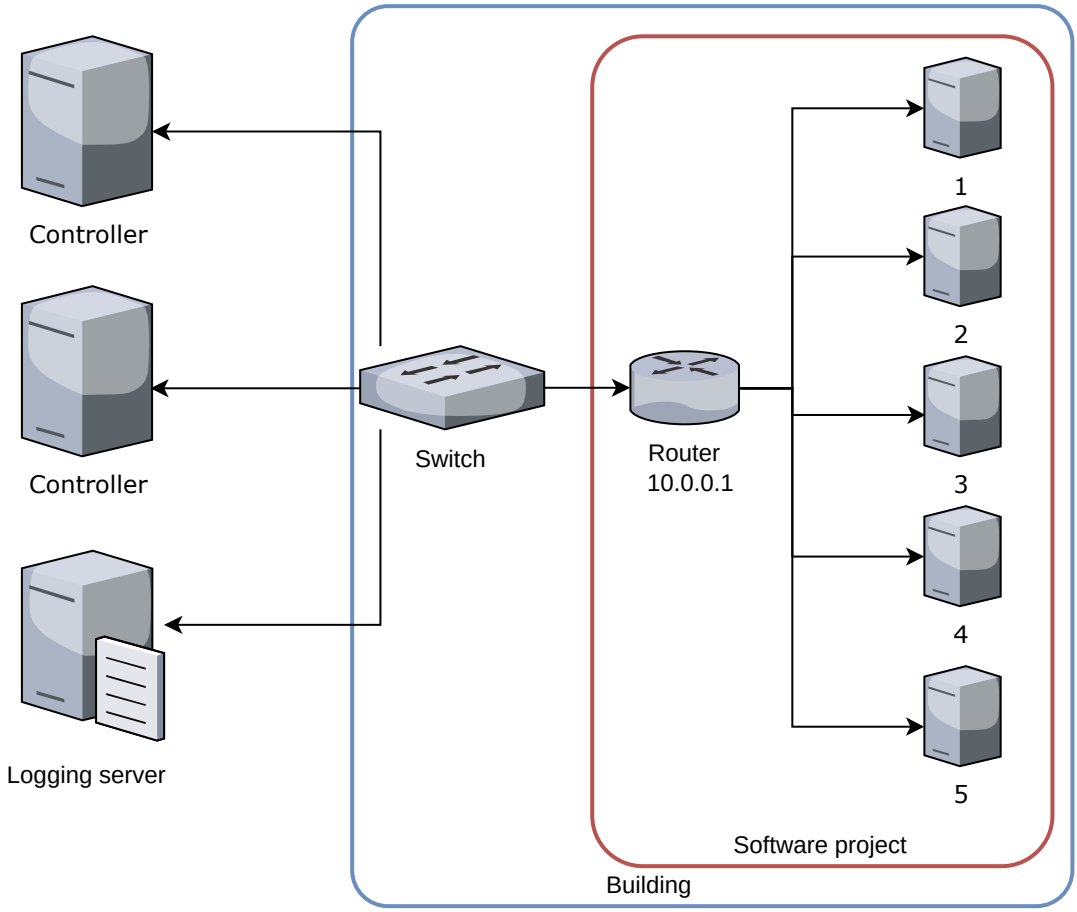

Figure 5: Network model.

4.2 Traffic features

Numerous features are available for each packet captured from the bus. An overview of a number of features is presented in Table 2 with a focus on BACnet related features originating from BACnet's Virtual Link Control, Network and Application layers. Each feature is listed together with its Wireshark key.

Table 2: Traffic features.

Feature name	**Feature key**
Time epoch Destination MAC Source MAC Destination IP Destination port Source IP Source port	frame.time_epoch eth.dst eth.src ip.dst ip.src udp.srcport udp.dstport
BVLC BACnet type BVLC Function	bvlc.type bvlc.function
BACnet version Destination network Source network Destination address Source address Control Message type Hop Count	bacnet.version bacnet.dnet bacnet.snet bacnet.dadr_eth bacnet.sadr_eth bacnet.control bacnet.mesgtyp bacnet.hopc
PDU type Invoke ID Service Choice Unconfirmed Service Choice Object type Object instance Properties Vendor	bacapp.type bacapp.invoke_id bacapp.confirmed_service bacapp.unconfirmed_service bacapp.objectType bacapp.instance_number bacapp.property_identifier bacapp.vendor_identifier

4.3 Controller inspection

By inspecting traffic sent to the campus building controllers, the purpose of the sending controller can be determined by studying the types and patterns of messages sent.

4.3.1 Logging server (network historian)

On the university campus a dedicated logging server logs information for all BMSs. The historian's traffic is distinguishable by the fact that it exclusively reads TrengLog objects, as shown in Figure 6. For controllers 4 and 5 logs are kept by the historian. TrengLog properties are requested every hour of the day. At the end of each day at 23:00, the historian reads the TrendLog Logbuffer of each object – via the ReadRange service. The Logbuffer contain a list of periodic value changes with a timestamp (in the network capture, each time component was found to be all zero). The object's properties always return a 'Property Access Error'. Additionally, all TrendLog objects lack a monitored object reference.

4.3.2 Inter-controller traffic

There is at least one identifiable non-Priva BACnet controller active in the network. This device generates traffic of ReadPropertyMultiple to request the system-status property of many objects.

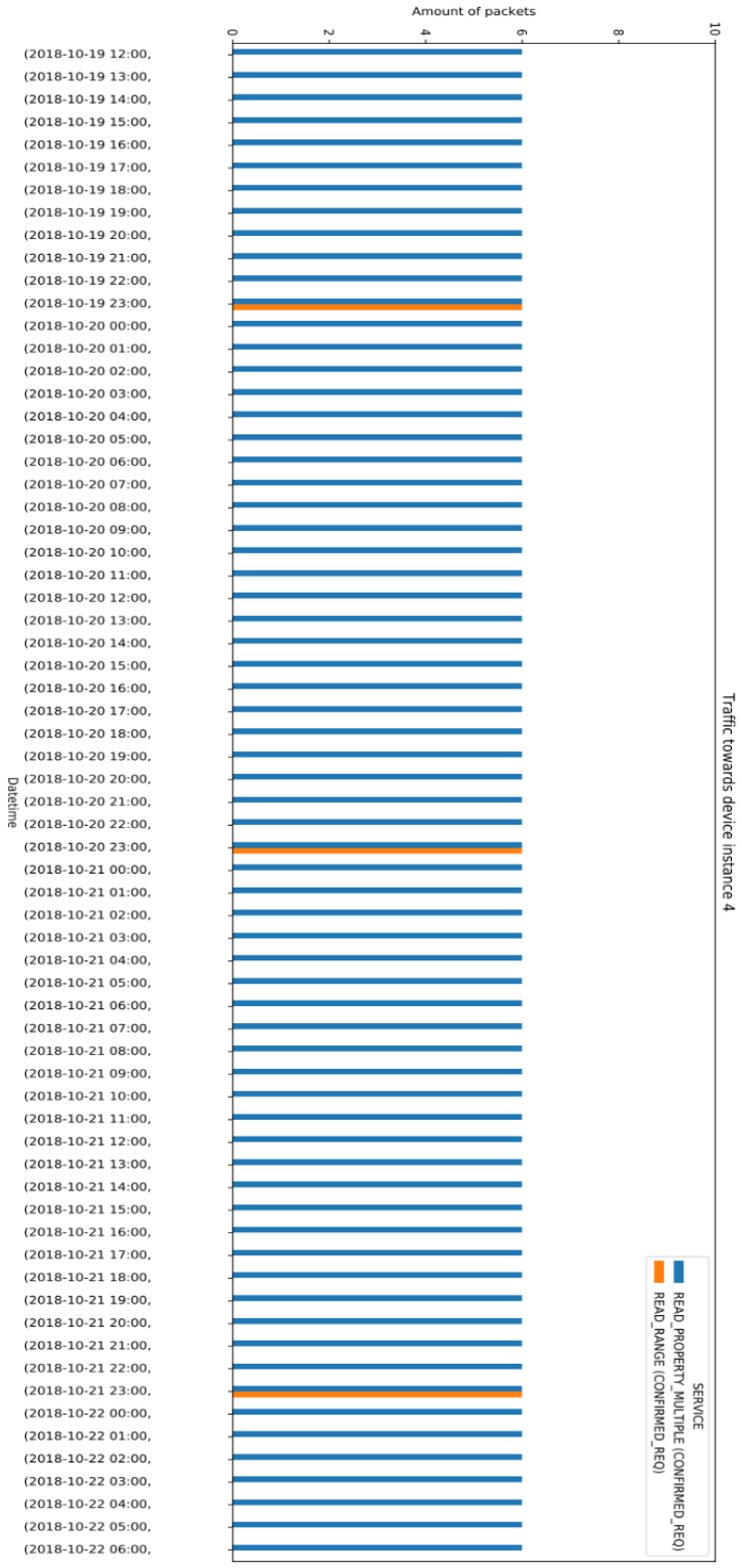

Figure 6: Outgoing traffic from logging server to BACnet Router at 10.0.0.1 towards device instance 4.

5. Problem statement

Preceding a targeted attack on a BMS, the threat agent will need to have sufficient knowledge about the BMS under attack. During his intelligence gathering phase, the agent will learn specifics of the BMS before creating an attack plan. In this thesis the focus is on this intelligence gathering phase, at which time an attacker has (limited) access to a BMS and wishes to get more details. The aim is to identify BACnet objects that together provide some functionality such that these can be carefully targeted.

Specifically, the goal is to find out what BACnet objects are related. It is assumed that the threat agent's only capability during the intelligence gathering phase is to passively monitor bus traffic in a stealthy manner to gather intelligence about the BMS.

Based on the problem statement, the following research questions come forward:

RQ How can functional relations among BACnet objects be identified from the bus traffic?

To answer the main research question the following sub-questions should be answered:

RQ-I How can a ground truth of all relations among object be established?
RQ-II How can object relations be extracted from bus traffic captured from a communications switch?

6. Establishing a ground truth of object relations (RQ-I)

For the purpose of evaluation a ground truth of all object relationships is necessary. The ground truth will define the de facto relationships among all objects at any software module level.

6.1 Source of the ground truth

For the creation of a ground truth a mapping must be made from objects identifiers to software modules. If multiple levels of subdivision are possible in the software tool, a clear definition of this hierarchy structure of subdivisions is required too. A logical place to look for this information is the software project file created by a software tool that was used to create and commission a BMS.

The BMS software project concerns itself with the building on the university campus. The part of the BMS described by this software project exists solely of Priva controllers and hence uses the same vendor's software tool 'TC Select'. The software tool utilizes 'PPF' files to store project data.

6.2 Reverse engineering the project file

All names in the database, tables and columns, are defined in Dutch. Column names follow a naming scheme from which it is easy to determine their type: the first character of a type is prefixed to the column name. Prefixes used are 'L' for long, 'B' for boolean, 'S' for string and 'M' for memo. Textual descriptions are stored in a separate table 'Teksten' each identified with its own id 'Lomschrid' and columns for multiple display languages.

To establish an initial understanding of the relation of tables to elements in the TC Select user interface, textual descriptions in the database tables have been cross referenced with textual descriptions found in the TC Select interface. Due to the absence of foreign keys in the database, table relations were determined manually by finding columns with names corresponding to a table's primary key. The foreign keys were either never defined or the database viewer did not display them.

After this a database structure could be created with all tables and relations relevant for understanding the software module hierarchy. Figure 7 shows all table names and primary keys as they were found in the project database. Each of the ElementTypen entries derives from a parent entry contained in an external "library" table, which has been omitted for simplicity.

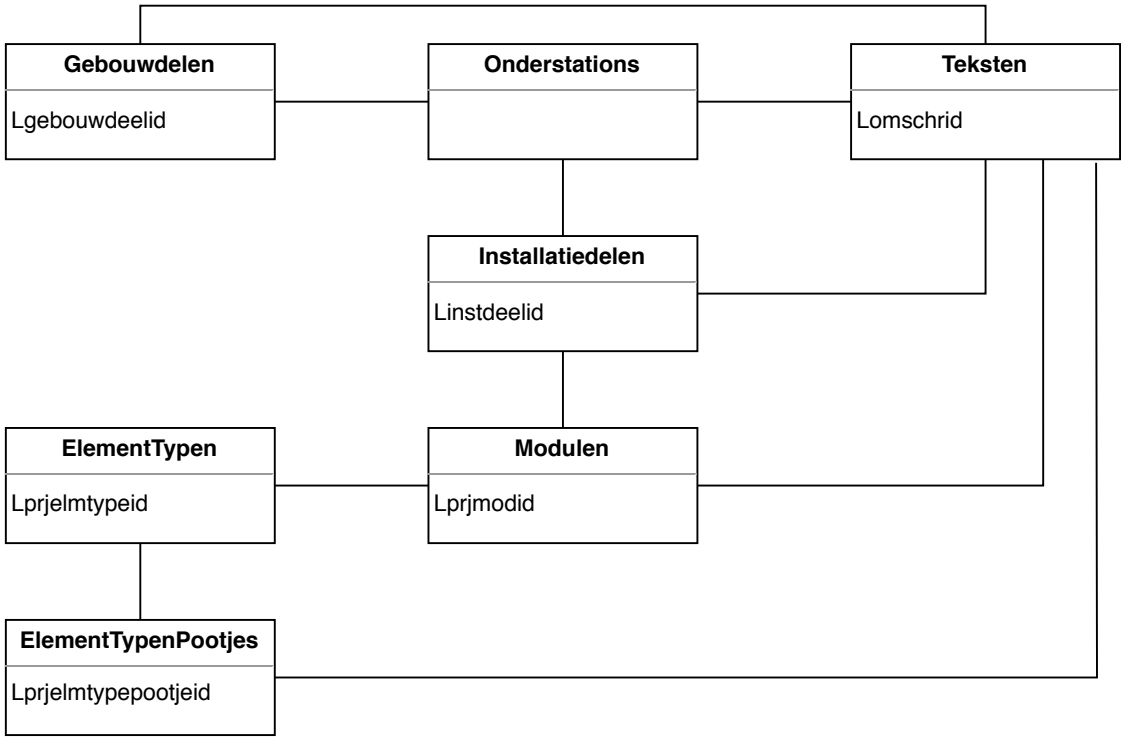

Figure 7: TC Select database structure.

Each ‘Onderstation’ represent a controller which has its own unique controller id “Londerstationid”. Based on their descriptions a mapping from controller id to the Device instance number was made (See Table 3). Each of these controllers is subordinated to controller id ‘31’ which identifies itself as “BACnet Router”. This is in accordance with the network environment as described in Section 4.1.

Table 3: Mapping from controller id to Device instance.

Londerstationid (Controller id)	**Device instance**	**Description**
1	1	RK1 (verlichting)
4	3	Lab RK2
10	4	Verwarming
12	2	RK2 TR
30	5	RK2 TCM

There is a clear hierarchical structure in the project database as shown in Figure 8. The hierarchy shows that a Priva BMS can contain multiple buildings which are subdivided into multiple sections, each of which has one or more controllers.

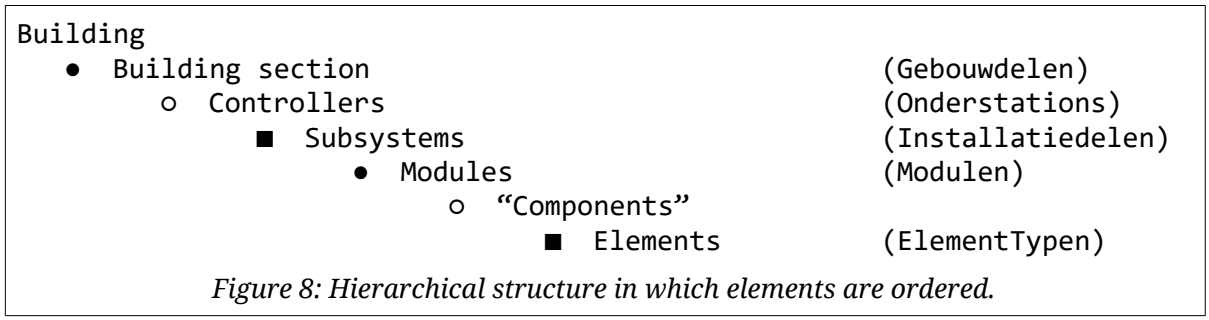

Figure 8: Hierarchical structure in which elements are ordered.

A controller manages one or more subsystems, each containing one or more modules. Finally, a module is made up of a multitude of elements. A group of elements of the same type work together to provide some kind of functionality. To facilitate talking about such groups of elements they are referred to as “component”.

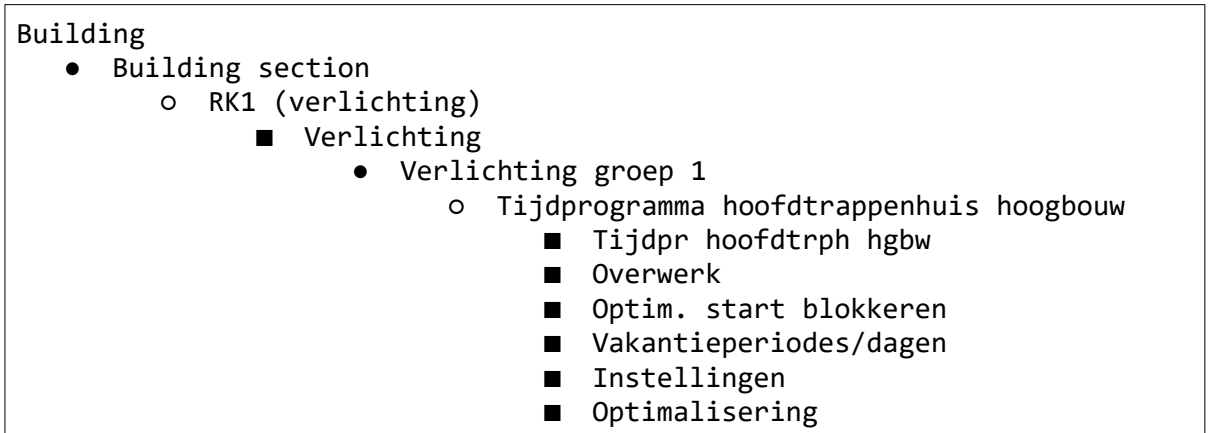

Figure 9: Hierarchical structure in which elements are ordered.

Elements are high-level constructs designed by Priva. More elements can be added to the software tool by installing additionally element libraries if desired. Not all elements necessarily map to one BACnet object: some map to none, some to one or more. Figure 9 shows an example for a given component ‘Tijdprogramma hoofdtrappenhuis hoogbouw’. This component can be seen in the graphical schema editor (Figure 4), but is broken down internally in many elements.

The information relevant for the ground truth, i.e. to what controller, subsystem and module an element belongs, is stored in a lookup table. Each entry in the lookup table is identified by a unique key composed of the controller id and the descriptions of the subsystem and controller as well as the description of the element. For the element “Tijdpr hoofdtrph hgbw” in Figure 9 the lookup key thus is “1-Verlichting/Verlichting groep 1/Tijdpr hoofdtrph hgbw”. The practical use of this lookup table is further discussed in Section 6.4.

6.3 Complementing the database

The Priva project database alone is not enough to create a ground truth. The database only shows how the software project is subdivided and where components reside in the project hierarchy. To learn what object identifiers are associated with what elements the Engineering Data Exchange (EDE) file for the project is to be consulted.

A plugin in TC Select is used to produce the needed EDE spreadsheet files. It should be noted that some of the columns in said EDE files are non-standard (e.g. they have not been adapted into the official EDE standard). The non-standard columns conveniently indicate the name of hierarchy levels “part of installation”, “module” and “element”. Using these non-standard columns an object identifier can be mapped to information from the project database and the ground truth is finished. Not all objects could be mapped. Two objects have been discarded because no suitable match was made (see Table 4 for precise statistics).

6.4 Ground truth in practice

For each incoming BACnet message the sending or receiving device instance (controller) is determined. Subsequently, all objects identifiers are extracted. Using the object identifier the EDE files are consulted for the names of each object’s corresponding “part of installation”, “module” and “element”. These names are used to query the lookup table created earlier from the project database. Once the lookup returns an entry the following properties are known for the given object:

- Device instance
- Object type
- Object instance number
- Subsystem name + unique id
- Module name + unique id
- Element identifiers

Table 4 shows a number of statistics for the devices in the BMS extracted from the ground truth. Some devices appear to have much more BACnet objects than others. The number of software modules (subsystems and modules) varies greatly among the controllers. For device instance 1 the two Device objects could not be mapped as such objects are not defined in the software project.

Table 4: Statistics extracted from the ground truth.

Device instance	**# of Objects (EDE)**	**# of objects (Mapped)**	**# of Subsystems**	**# of Modules**
1	**232**	**230**	7	18
2	1654	1654	12	44
3	1005	1005	18	27
4	499	499	8	12
5	1665	1665	9	30

6.5 Results

It has been shown that a ground truth containing object relations can be established based on the available BMS project file and EDE file. Using the database contained in the project file, software modules can be extracted for each element. Combining this information with data available in the EDE files yields a complete ground truth describing the software modules for each BACnet object in the project.

7. Discovery of object relations from bus traffic (RQ-II)

The goal of this step is to gather BACnet objects from the bus traffic and group them into their respective software modules. As per the definition of object relations, objects in the same software module are related. Note that the aim is not to record directional inter-object relations, but rather a high-level grouping of related objects into software modules. For the method to be successful, a threat agent should ultimately be able to perform the chosen method using unknown bus traffic. Therefore, it is deemed important for the method to be repeatable and generic in nature (i.e. not specifically trained for use on a single data set).

In the field of data mining there are many methods concerned with the extraction of knowledge from data. Similarly, there are methods providing insight into relationships between data points. The following two steps can be distinguished when using a data mining method:

1) Learning and training stage. At this time all access to information about the BMS is warranted for the sake of model creation.

2) Testing stage. The created model is applied on a new and unknown (part of a) BMS. The result should give insight into how objects are related.

To achieve the desired repeatability and generality an appropriate method should satisfy the following requirements:

- Division of related data points into logical groups.
- Classification of data into relative groups (i.e. it should not assign – previously learned – absolute class labels to data points).

7.1 Clustering: a naive approach to object relations

Unsupervised data mining algorithms fulfill the requirements listed in the previous section and as a result clustering was selected as the method of choice. Clustering divides a set of points into some number of clusters of which all members are similar in some sense and where members of different clusters are dissimilar. The similarity between members of a cluster is based on the distance between them. There are at least a number of clustering algorithms that can deal with categorical variables such as k-modes, k-medoids and DBSCAN (Density Based Spatial Clustering of Applications with Noise).

Inherent to unsupervised algorithms is the fact that data is unlabeled and the process of classification is performed without human interference. It is, however, often the case that there is indeed some prior knowledge available about the input data. In the context of this research, there is prior knowledge about relations that are present based on the protocol semantics. To encode this knowledge into the clustering algorithm, meaningful distance measures are introduced for the data set. The ideal distance measure must incorporate detailed knowledge about protocol semantics to establish point locations in space. An implementation of the DBSCAN algorithm with

support for variable type distance metrics called ‘TypeCluster’ is available. DBSCAN is density-based and therefore, in contrast to k-medoids, there is no need to specify the number of clusters beforehand. Additionally, DBSCAN allows the grouping of data into arbitrarily shaped clusters. Therefore DBSCAN will be used as clustering algorithm in the upcoming steps. To evaluate the partitions made by the clustering algorithm, the true labels from the ground truth are used. It should be clear that these true labels are not used as feature.

Table 5 shows a road map containing all steps that are needed for this approach based on the unified framework for knowledge discovery in databases (KDD) by Fayyad et al. [17]. Fayyad et al. describe the KDD process as a set of activities of which data mining is a crucial part. The process includes the enumeration of patterns (model creation) and the evaluation of these patterns to find a subset of patterns deemed “knowledge”. It is emphasized that the KDD process is iterative in nature and there may be loops between any steps. The steps will be further discussed in the remainder of this section.

Table 5: Clustering road map.

<table>
<tr><th>Step</th><th colspan="2">Activity</th><th colspan="2">Output</th></tr>
<tr><td>1</td><td colspan="2">Selection – selecting a subset of the data set and variables that are available</td><td colspan="2">Target data</td></tr>
<tr><td>2</td><td colspan="2">Pre-processing – remove noise and handling of incomplete/missing variables and creation of composite variables</td><td colspan="2">Pre-processed data</td></tr>
<tr><td>3</td><td colspan="2">Transformation – data dimensionality reduction through feature selection</td><td colspan="2">Transformed data</td></tr>
<tr><td rowspan="2">4</td><td rowspan="2">Model creation - iterative mining and evaluation to select optimal parameterization of method</td><td>4a Data mining – application of model and parameters</td><td>Clustered data</td><td rowspan="2">Clustering model</td></tr>
<tr><td>4b Evaluation – evaluation of the clustering result</td><td>Evaluation score</td></tr>
</table>

7.1.1 Selection

The supplied BMS bus traffic captures contain all traffic going through the building's core switch. Since only a ground truth is available for a selection of the BMS, the captures are filtered to only contain traffic going from and to this part of the BMS. The selection encompasses traffic starting at 14:22, 19 October 2018 through 11:20, 24 October 2018. A first feature selection is made which includes time related as well as many NPDU and APDU features.

7.1.2 Preprocessing

The traffic capture is first preprocessed to extract all BACnet objects. This step is implemented in the form of a Java application. The application takes the selected data from the previous step and carries out the following steps in sequence:

- Assign each packet with its corresponding device instance based on observed "I-am" packets (this allows unique mapping from a BACnet address to a device instance) based on the method described by Caselli et al. [18].
- Assign each packet with its corresponding controller id (extracted from project database) based on its device instance. This process is based on a prior-knowledge of the mapping from device instance numbers to controllers in the ground truth.
- Set missing feature values to 0.
- Assign each packet with the following evaluation labels from the ground truth based on its controller id and object identifier:
    a. Subsystem id.
    b. Module id.
- For each packet extract all object identifiers and save them.
- Objects are split into different output files based on controller id. By doing so the objects are grouped into their corresponding software module on a coarse level.

Any objects for which the ground truth is unavailable are discarded from the data set. For example, when a packet's controller id is unknown or there is are no suitable mappings to entries in the ground truth for the objects contained in said packet, it will be removed. From a total of 5229 unique objects extracted from the bus traffic, 209 were discarded.

7.1.3 Transformation

The processed data from step 2 is used to carry out feature selection. Two feature selection algorithms were used and their output was compared as to rule out any noticeable discrepancies.

The information gain ratio algorithm from the 'R' FSelectorRcpp package is used to determine information gain ratio for features in the data set in correlation to each of the possible response variable vectors. The response variable vectors are taken from the ground truth directly and correspond to the different software modules subsystem and module.

The second feature selection algorithm used is Boruta from the eponymous R package. Boruta is a so called wrapper algorithm which derives its core functionality

from random forest classification. Boruta uses an all-relevant feature selection algorithm as opposed to many other feature selection algorithms which use a minimal-optimal method. This means that Boruta aims to find all features important for the classification instead of a more compact set carrying just enough information to perform an optimal classification. Boruta selects features by adding more randomness into the system. Each iteration, a copy of part of the original feature space is merged with randomized shadow variables after which a random forest classifier is trained on the system. For each of the features in the trained random forest models the Z-score (dividing average accuracy loss by its standard deviation) is calculated. Boruta uses the Z-score as importance measure. Using these Z-scores for both the real and shadow features, each of the real features can be tagged as either important or unimportant. There is closure as soon as all features have been tagged.

Table 6 shows the features that are deemed important by both information gain as well as Boruta.

Table 6: Features deemed important by Boruta and information gain ratio.

Feature
Time epoch
Service Choice
Unconfirmed Service Choice
Object type
Object instance
Invoke ID
Properties
PDU type
Control
Destination MAC
Destination IP

Table 7: Features after feature reduction

Feature
Time epoch
Object type
Object instance
Properties

After test runs of the DBSCAN algorithm with these features more of the features in Table 6 are discarded. A heavy emphasis is laid on timing as this is believed to play a key role in finding relationships. The remaining features are shown in Table 7.

7.1.4 Model creation

For model creation a Python application was developed using TypeCluster in combination with the DBSCAN implementation of Scikit learn[1]. The DBSCAN algorithm has several parameters that control the clustering operation. DBSCAN determines clusters based on epsilon 'Ɛ' and n, where Ɛ is the maximum distance between two points in the same cluster and *n* the minimum number of points in a single cluster. During model creation the goal is to find a parameterization of the DBSCAN algorithm for which it is able to group BACnet objects into software modules.

7.1.4.1 Distance metrics

In practice, TypeCluster acts as a DBSCAN wrapper with the ability to compute distance matrices for a wide variety of weighted distance functions using the Minkowski distance. The Minkowski distance between points x and y is defined as:

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

The Minkowski distance is a metric in a normed vector space (i.e. distances are defined using vectors with certain length) which generalizes Manhattan (p = 1), Euclidean (p = 2) and Chebyshev (p = ∞) distances. This allows many different distance metrics to be plugged-in into TypeCluster, which on its turn generates the necessary distance matrix for the clustering algorithm.

To compute the distance between two data rows, in this case being BACnet objects with the selected features, each feature's distance metric is invoked and the weighted distance of each features counts towards the final distance (i.e. the distance between the object). A distance metric takes two arguments, *u* and *v*, and calculates the distance between them based on its implementation. Four different distance metrics are used:

1) Discrete, binary Y/N decisions.
2) Jaccard, array comparison.
3) Levenshtein, string comparison.
4) Object distance, object identifier comparison.

In Table 8 an overview is given of all features and their corresponding distance metric. Each feature has been assigned to one of three categories indicating its type. A distinction between object properties, traffic behavior and external properties is made. The first two features are decidedly very objective properties of a BACnet object. The next four features describe how the object behaves in the bus traffic. The last feature, 'Graph id' is categorized as external property, as it was derived from an external source, i.e. previous research.

1 https://scikit-learn.org/0.17/modules/generated/sklearn.cluster.DBSCAN.html

Table 8: Features and their distance metrics.

Feature	Distance metric	Category
Object identifier (composite of object type and instance number)	"Object distance" (if equal type, distance is set to 0 otherwise take difference of instance number)	BACnet object property
Description	Levenshtein	
Property identifier array	Jaccard	Object traffic behavior
Epoch difference array	Jaccard	
Time of day array	Jaccard	
Packet common objects	Jaccard	
Graph id	Discrete	External object property

7.1.4.2 Evaluation criteria

A clustering result is evaluated based on three criteria. The main criterium is an information theoretic measure called Adjusted Mutual Information (AMI) score. AMI is a normalization of Mutual Information (MI) and set to be 0 when two partitions of a set, the clustering and the set of true labels, are distributed randomly. Similarly, AMI is 1 when the distributions of partitions are identical. Furthermore, for each clustering the Homogeneity (Hg) and Completeness (Cp) are calculated. Homogeneity is satisfied when all clusters contain only data point from a single software module. Completeness is satisfied when all data points from a single software module are contained in the same cluster. All scores have positive values between 0 and 1.0, larger values indicate a better result.

7.1.4.3 Python program

The Python code for model creation performs the following steps:

1) Initialize index *i* to 0, Initialize weight vector V_w to all ones.
2) Calculate the distance matrix with the V_w.
3) Use distance matrix as input for DBSCAN and repeat with parameter ε ranging from ε-min through ε-max. Return the parameterization of the clustering with highest AMI score.
4) Compare the returned score to that of the last:
    a) If current is better: increase the weight of $V_w[i]$ by 10.
    b) If equal: do nothing.
    c) Else: decrease weight of $V_w[i]$ with 5.
5) Increase *i* with 1 (modulo the amount of features) and repeat from step 2.

7.2 Results

In this section a summary of the results of the model creation step in Section 7.1.4 is shown. Table 9 lists the clustering results of the first ten runs of the model creation algorithm for device instance 2. For each run the model parameterization (ε and corresponding weight vector) is shown together with evaluation criteria AMI, Homogeneity (Hg) and Completeness (Cp). Finally, the number of clusters estimated by the clustering algorithm is given. A column for the amount of noise points has been omitted as this was recorded to be zero in all cases. For each controller, the first runs with the lowest and highest AMI score have been emphasized.

The algorithm has been run for 55 iterations for each device instance. Please refer to Appendix A for the unsummarized results in Tables A-1 through A-5 showing results for device instances 1 through 5 respectively. For a discussion of the results, see Section 7.3.

Table 9: First ten clustering results for controller with device instance 2.

run	ε	clusters	AMI	Hg	Cp	V_w
1	17	178	.094	.287	.199	[1, 1, 1, 1, 1, 1, 1]
2	**15**	**247**	**.086**	**.386**	**.195**	**[11, 1, 1, 1, 1, 1, 1]**
3	45	292	.095	.443	.211	[6, 11, 1, 1, 1, 1, 1]
4	64	282	.098	.437	.212	[6, 21, 1, 1, 1, 1, 1]
5	75	281	.098	.437	.212	[6, 31, 1, 1, 1, 1, 1]
6	89	278	.099	.437	.212	[6, 41, 1, 1, 1, 1, 1]
7	99	274	.100	.434	.211	[6, 51, 1, 1, 1, 1, 1]
8	108	270	.101	.434	.212	[6, 61, 1, 1, 1, 1, 1]
9	**116**	**269**	**.101**	**.433**	**.211**	**[6, 71, 1, 1, 1, 1, 1]**
10	122	268	.101	.432	.211	[6, 81, 1, 1, 1, 1, 1]

Figures 10 and 11 show a graphical representation of the results for device instance 2. Figure 10 shows the three evaluation criteria AMI, Hg and Cp plotted against parameter ε. In this plot more than one point may appear on the Y-axis for the same value of ε on the X-axis. Figure 11 shows the amount of clusters plotted against ε. The Figures for each device instance can be found in Appendix A. Results. In Table 10 a summary of a selected statistics and evaluation criteria is presented for each controller's best scoring run.

Controller 2

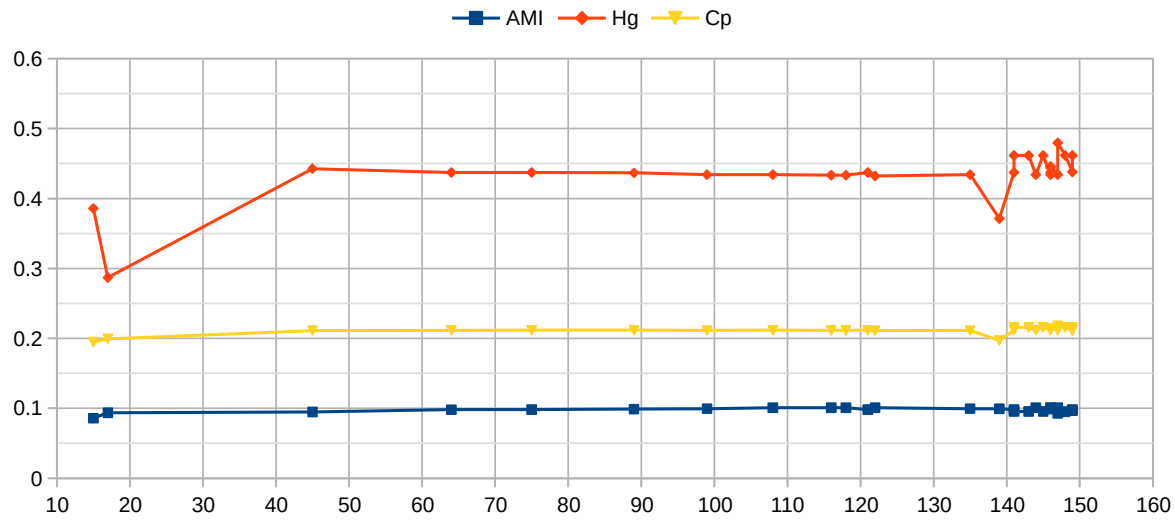

Figure 10: AMI, Homogeneity and Completeness against ε for controller with device instance 2.

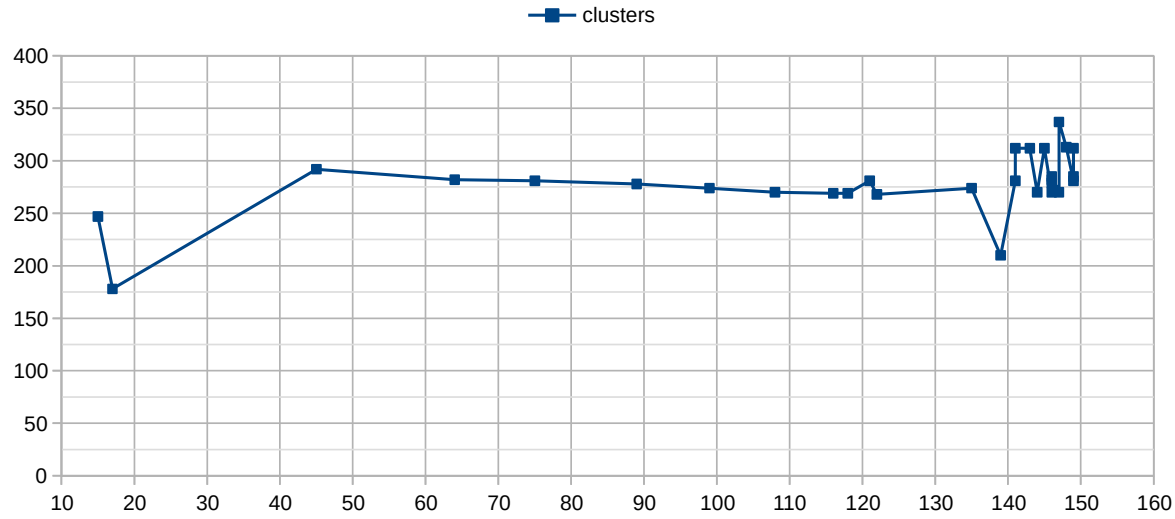

Figure 11: Amount of clusters against ε for controller with device instance 2.

Table 10: Summary of controller statistics and "best run" evaluation criteria

Device instance	**# of Objects**	**# of Subsystems**	**ε**	**AMI score**	**# of clusters**	**V_w**
1	231	7	25	.448	14	[31, 1, 1, 1, 1, 1, 1]
2	1654	12	116	.101	269	[6, 71, 1, 1, 1, 1, 1]
3	1005	18	119	.111	167	[6, 51, 1, 1, 1, 1, 1]
4	498	8	109	.223	76	[11, 41, 11, 11, 11, 21, 16]
5	1665	9	61	.117	46	[6, 11, 1, 1, 1, 1, 1]

7.3 Discussion

Comparing the scores of the controllers among each other, the best scoring controller is 1 (See Table 10). The controllers with fewer objects, 1 and 4, outperformed the controllers with a large number of objects (over 1000). For the latter group of controllers scores range from 0.101 to 0.117. The estimated number of clusters for controller 1, 14, is very close to the actual number of subsystems in the ground truth: 7. For the remaining controllers, the discrepancy between estimated clusters and actual amount of subsystems is far larger.

In Figures B-1 through B-5 it can be seen that the shape of the homogeneity (Hg) curve roughly follows the shape of the curve of the amount of clusters. A greater amount of clusters positively influences homogeneity as a single cluster will contain fewer points of which more are of the same software module.

Looking at AMI and completeness (Cp), in Figure A-1, there is a relatively large change for both the completeness and AMI score accompanied by a steep drop in the number of clusters at ε values between 10 and 30. In Figure B-3 there are clear value changes for the evaluation criteria as well, for ε between 90 and 120. As the number of clusters increases dramatically, so do Hg and Cp, but the AMI score drops. The increase of Hg is directly tied to the increased number of clusters. The improved Cp is caused by an increased number of clusters with more objects belonging to the same software module (i.e. objects in the same software module are spread among less clusters). Surprisingly, the lower AMI score means that the overall distribution of objects over clusters cannot be distinguished from a random distribution any better. In fact it is even slightly harder to tell apart.

Interestingly, in Figures B-2, B-4 and B-5, AMI and Cp follow a very flat trajectory for all other controllers. There is no visible improvement or deterioration of both scores.

From the results as shown in the tables in Appendix A and Figures in Appendix B it can be observed that the evaluation criteria for a given ε together with different weight vectors V_w do not yield very different results. In most cases, the evaluation criteria even stay exactly the same. This illustrates the limited effect the weighted distance metrics have on the final results.

Looking at features and distance metrics that have been used, it is noticeable that features in the category object properties are generally weighed higher than others. Only controller 4 differentiates itself from the rest; all features are weighed higher and Description and Packet common objects being outliers. The resulting AMI score for 4 is nevertheless very low so this does hardly say anything about the actual usefulness of said features. In practice the graph id features turns out to be of less for some controllers as many objects could not be assigned a meaningful graph id.

Studying the results leads to believe that the metrics for features in the BACnet object property category indeed are more suited to controllers 1 and 4. The assumption for the description's distance metric is that objects in different modules have a unique name or include an identifier indicating the module they are in. This assumption does not hold for many modules in the other controllers as many modules are reused without any change to their object description. Based on the description metric, these objects would all belong to the same module. These kinds of differences in controller implementation make it very difficult to fit a single model that works well on all controllers.

8. Limitations

8.1 Ground truth

When it comes to the ground truth that was established, it should be noted that this method is only applicable to BMSs that were constructed using Priva's TC Select application. The non-standard EDE files generated by the same application contributed to the success of creating the ground truth. It cannot be expected that other application similarly include the names of software module in their EDE export. Although similar methods may exist to create a ground truth for other vendor's software applications these are not covered in this thesis.

8.2 Clustering

Key points during the thesis were the availability of captured network bus traffic for a BMS as well as the software project describing the same BMS. While bus traffic and a software project are both available, a limiting factor is the fact that the building's switch with the network tap, mainly captured inter-building traffic and did not show any inter-controller traffic among the controllers defined in the software project. The network model as seen in Figure 5 shows very clearly that all inter-controller traffic for these controllers is handled by another device, a router. Had the tap been placed in the router, if possible at all, the method presented in this thesis might have been more effective. The desired network model is depicted in Figure 12. In this case, all inter-controller traffic can be captured. It is though that as all Priva controllers are

defined in the same software project they all have exact knowledge of each other functions and tasks. Considering this, the network traffic between these controllers is likely to convey more information about software modules in the software project. In the current situation, controllers have limited knowledge about the object present in the Priva controllers and mostly perform read actions or subscribe to value changes. The traffic generated from these actions does not convey much meaning about the software modules, as the results show.

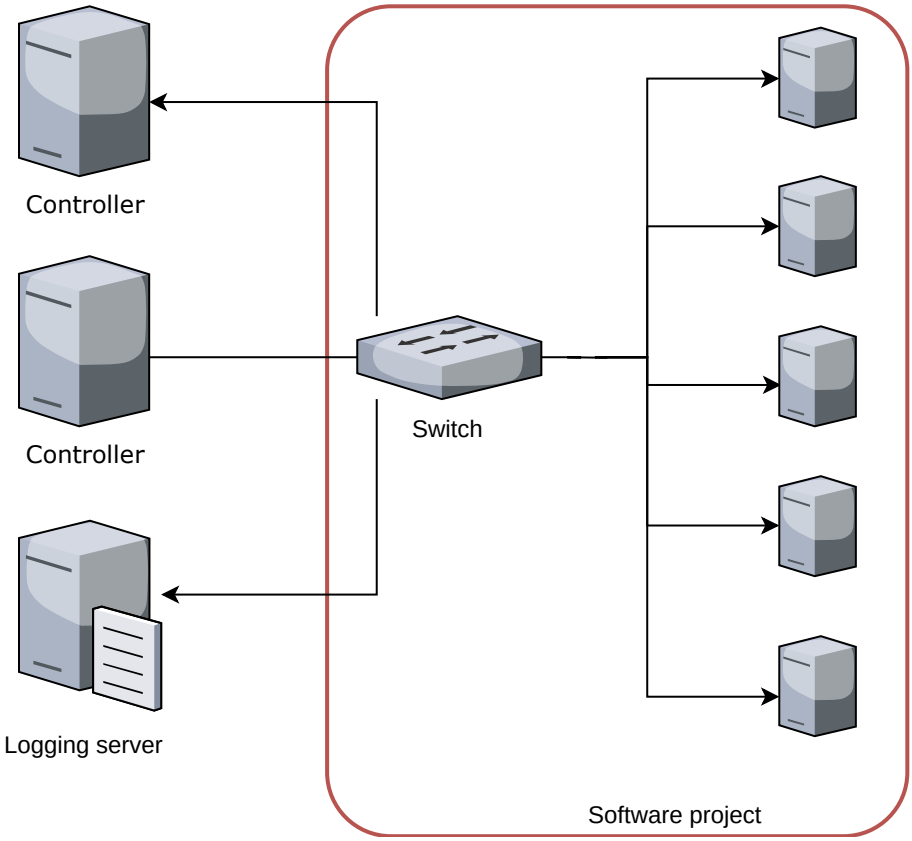

Figure 12: Desired network model.

9. Conclusion

In this paper a method is presented to create a ground truth from a BMS and extract BACnet object relations from bus traffic. A ground truth was successfully created from a software project that was used to commission a BMS. The method for object relations extraction uses unsupervised machine learning techniques, specifically the DBSCAN clustering algorithm. The clustering algorithm is so far unable to create satisfactory clusterings as evaluated using the AMI score. The results can partly be explained by the lack of good distance metrics. On the other hand, the data made available is captured in a less than favorable network setup. Gathering information about building specific controllers and their objects was perceived to be very difficult from the supplied data. Further research should focus on the creation of more suitable distance metrics that are able to capture object relations better. Ideally, such research should be performed in a BMS or experimental setup adhering to the desired network model. The usage of TrendLog objects looks promising and should be used as prior knowledge that could strengthen the clustering operation.

10. Further research

An interesting research direction for future work is on the subject of BACnet TrendLog objects. TrendLog objects reference an object for which trend logs are collected. The TrendLog object can be configured to subscribe to or poll said object for value changes are set intervals and stores the values in an internal record buffer. As TrendLogs store referenced values, upon collection of enough of these values one might be able to tell which of the objects monitored in trend logs are related based on value changes. To test this hypothesis an experimental setup modeling several software modules would need to be constructed. Using a method to find relations in multivariate time series data, related object should be able to be identified. Using this limited set of related objects, the method using a clustering algorithm should be enforced to improve results.

11. References

1. Hicks, Art. “Understanding Building Automation and Control Systems.” KMC Controls, web.archive.org/web/20130519124213/http://www.kmccontrols.com/products/Understanding_Building_Automation_and_Control_Systems.aspx.
2. Smith, A. “Building Management Systems (BMS), Seminar 1 - The Basics Explained.” City of Melbourne
3. GICSP, CISSP, CEH , Ernie Hayden, et al. “An Abbreviated Histor-357,-31y of Automation & Industrial Controls Systems and Cybersecurity.” Aug. 2014.
4. Contemporary Controls, George Thomas. “Raising BACnet® to the Next Level.” 2009, www.ccontrols.com/pdf/RaisingBACnetWP.pdf.
5. Hollinger, Christopher A. “Strategies for Using BACnet®/IP.” ASHRAE Journal, Oct. 2004, pp. S10–S12.
6. Michael Phillips. “Understanding Networks in Modern Building Automation Systems.” HPAC Engineering, 31 Oct. 2017, www.hpac.com/managing-facilities/understanding-networks-modern-building-automation-systems.
7. Peacock, Matthew, Michael N. Johnstone, and Craig Valli. "An exploration of some security issues within the BACnet protocol." International Conference on Information Systems Security and Privacy. Springer, Cham, 2017.
8. Fisher D, Isler B. and Osborne M. “BACnet Secure Connect - A Secure Infrastructure for Building Automation” AHRAE BACnet whitepaper. 21 May 2019.
9. SANS ICS Amsterdam Summit & Training 2014 (September 2014) Steffen Wendzel Newman, H. Michael. BACnet: the Global Standard for Building Automation and Control Networks. Momentum Press, 2013.
10. “Targeted Attacks.” Definition - Trend Micro USA, www.trendmicro.com/vinfo/us/security/definition/targeted-attacks.
11. Contemporary Controls, George Thomas. “Raising BACnet® to the Next Level.” 2009, www.ccontrols.com/pdf/RaisingBACnetWP.pdf.
12. BACnet/IP. http://www.bacnet.org/Tutorial/BACnetIP/index.html.
13. Merz, Hermann, Thomas Hansemann, and Christof Hübner. Building automation. Springer, 2009.
14. Schneider Electric, David Fisher. Capability to Import a BACnet Device Objects via an EDE File? 17 Sept. 2018, community.exchange.se.com/t5/Knowledge-Base/Capability-to-import-a-BACnet-device-objects-via-an-EDE-file/ta-p/3189.
15. BACnet Interest Group Europe. “Engineering Data Exchange Template for BACnet Systems ‘Description of the EDE Data Fields‘ Version of Layout: 2.3.” 16 Jan. 2017.
16. Yourdon E. and Constatine L. “Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design” Prentice Hall. 1979.
17. Fayyad, Usama M., Gregory Piatetsky-Shapiro, and Padhraic Smyth. "Knowledge Discovery and Data Mining: Towards a Unifying Framework." KDD. Vol. 96. 1996.
18. Caselli, Marco, et al. "Specification mining for intrusion detection in networked control systems." 25th {USENIX} Security Symposium ({USENIX} Security 16). 2016.

12. Appendices

Appendix A. Results

Table A-1: Clustering results for controller with device instance 1.

run	ε	clusters	AMI	Hg	Cp	V_w
1	19	28	.362	.734	.428	[1, 1, 1, 1, 1, 1, 1]
2	23	15	.373	.540	.422	[11, 1, 1, 1, 1, 1, 1]
3	24	14	.448	.540	.492	[21, 1, 1, 1, 1, 1, 1]
4	**25**	**13**	**.484**	**.524**	**.534**	**[31, 1, 1, 1, 1, 1, 1]**
5	25	13	.484	.524	.534	[41, 1, 1, 1, 1, 1, 1]
6	73	15	.373	.540	.422	[36, 11, 1, 1, 1, 1, 1]
7	54	15	.373	.540	.422	[36, 6, 11, 1, 1, 1, 1]
8	54	15	.373	.540	.422	[36, 6, 6, 11, 1, 1, 1]
9	54	15	.373	.540	.422	[36, 6, 6, 6, 11, 1, 1]
10	54	15	.373	.540	.422	[36, 6, 6, 6, 6, 11, 1]
11	54	15	.373	.540	.422	[36, 6, 6, 6, 6, 6, 11]
12	54	15	.373	.540	.422	[46, 6, 6, 6, 6, 6, 6]
13	**81**	**23**	**.360**	**.660**	**.421**	**[41, 16, 6, 6, 6, 6, 6]**
14	74	15	.373	.540	.422	[41, 11, 16, 6, 6, 6, 6]
15	74	15	.373	.540	.422	[41, 11, 26, 6, 6, 6, 6]
16	74	15	.373	.540	.422	[41, 11, 21, 16, 6, 6, 6]
17	74	15	.373	.540	.422	[41, 11, 21, 11, 16, 6, 6]
18	74	15	.373	.540	.422	[41, 11, 21, 11, 11, 16, 6]
19	74	15	.373	.540	.422	[41, 11, 21, 11, 11, 11, 16]
20	74	15	.373	.540	.422	[51, 11, 21, 11, 11, 11, 11]
21	78	33	.362	.807	.432	[46, 21, 21, 11, 11, 11, 11]
22	81	23	.360	.660	.421	[46, 16, 31, 11, 11, 11, 11]
23	81	23	.360	.660	.421	[46, 16, 26, 21, 11, 11, 11]
24	81	23	.360	.660	.421	[46, 16, 26, 16, 21, 11, 11]
25	81	23	.360	.660	.421	[46, 16, 26, 16, 16, 21, 11]
26	81	23	.360	.660	.421	[46, 16, 26, 16, 16, 16, 21]
27	89	15	.373	.540	.422	[56, 16, 26, 16, 16, 16, 16]
28	89	15	.373	.540	.422	[66, 16, 26, 16, 16, 16, 16]
29	103	23	.360	.660	.421	[61, 26, 26, 16, 16, 16, 16]
30	92	23	.360	.660	.421	[61, 21, 36, 16, 16, 16, 16]
31	92	23	.360	.660	.421	[61, 21, 31, 26, 16, 16, 16]
32	92	23	.360	.660	.421	[61, 21, 31, 21, 26, 16, 16]
33	92	23	.360	.660	.421	[61, 21, 31, 21, 21, 26, 16]
34	92	23	.360	.660	.421	[61, 21, 31, 21, 21, 21, 26]
35	101	15	.373	.540	.422	[71, 21, 31, 21, 21, 21, 21]
36	101	15	.373	.540	.422	[81, 21, 31, 21, 21, 21, 21]
37	117	22	.365	.658	.424	[76, 31, 31, 21, 21, 21, 21]

38	87	33	.362	.807	.432	[76, 26, 41, 21, 21, 21, 21]
39	87	33	.362	.807	.432	[76, 26, 36, 31, 21, 21, 21]
40	87	33	.362	.807	.432	[76, 26, 36, 26, 31, 21, 21]
41	87	33	.362	.807	.432	[76, 26, 36, 26, 26, 31, 21]
42	87	33	.362	.807	.432	[76, 26, 36, 26, 26, 26, 31]
43	103	24	.365	.681	.426	[86, 26, 36, 26, 26, 26, 26]
44	113	15	.373	.540	.422	[96, 26, 36, 26, 26, 26, 26]
45	113	15	.373	.540	.422	[106, 26, 36, 26, 26, 26, 26]
46	121	23	.360	.660	.421	[101, 36, 36, 26, 26, 26, 26]
47	117	22	.365	.658	.424	[101, 31, 46, 26, 26, 26, 26]
48	117	22	.365	.658	.424	[101, 31, 56, 26, 26, 26, 26]
49	117	22	.365	.658	.424	[101, 31, 51, 36, 26, 26, 26]
50	117	22	.365	.658	.424	[101, 31, 51, 31, 36, 26, 26]
51	117	22	.365	.658	.424	[101, 31, 51, 31, 31, 36, 26]
52	117	22	.365	.658	.424	[101, 31, 51, 31, 31, 31, 36]
53	123	15	.373	.540	.422	[111, 31, 51, 31, 31, 31, 31]
54	123	15	.373	.540	.422	[121, 31, 51, 31, 31, 31, 31]
55	109	33	.362	.807	.432	[116, 41, 51, 31, 31, 31, 31]

Table A-2: Clustering results for controller with device instance 2.

run	**ε**	**clusters**	**AMI**	**Hg**	**Cp**	**V_w**
1	17	178	.094	.287	.199	[1, 1, 1, 1, 1, 1, 1]
2	**15**	**247**	**.086**	**.386**	**.195**	**[11, 1, 1, 1, 1, 1, 1]**
3	45	292	.095	.443	.211	[6, 11, 1, 1, 1, 1, 1]
4	64	282	.098	.437	.212	[6, 21, 1, 1, 1, 1, 1]
5	75	281	.098	.437	.212	[6, 31, 1, 1, 1, 1, 1]
6	89	278	.099	.437	.212	[6, 41, 1, 1, 1, 1, 1]
7	99	274	.100	.434	.211	[6, 51, 1, 1, 1, 1, 1]
8	108	270	.101	.434	.212	[6, 61, 1, 1, 1, 1, 1]
9	**116**	**269**	**.101**	**.433**	**.211**	**[6, 71, 1, 1, 1, 1, 1]**
10	122	268	.101	.432	.211	[6, 81, 1, 1, 1, 1, 1]
11	118	269	.101	.433	.211	[6, 76, 11, 1, 1, 1, 1]
12	118	269	.101	.433	.211	[6, 76, 21, 1, 1, 1, 1]
13	118	269	.101	.433	.211	[6, 76, 16, 11, 1, 1, 1]
14	118	269	.101	.433	.211	[6, 76, 16, 6, 11, 1, 1]
15	118	269	.101	.433	.211	[6, 76, 16, 6, 6, 11, 1]
16	118	269	.101	.433	.211	[6, 76, 16, 6, 6, 6, 11]
17	121	281	.098	.437	.212	[16, 76, 16, 6, 6, 6, 6]
18	139	210	.099	.371	.197	[11, 86, 16, 6, 6, 6, 6]
19	135	274	.100	.434	.211	[11, 96, 16, 6, 6, 6, 6]
20	144	270	.101	.434	.212	[11, 106, 16, 6, 6, 6, 6]

21	147	270	.101	.434	.212	[11, 116, 16, 6, 6, 6, 6]
22	146	270	.101	.434	.212	[11, 111, 26, 6, 6, 6, 6]
23	146	270	.101	.434	.212	[11, 111, 21, 16, 6, 6, 6]
24	146	270	.101	.434	.212	[11, 111, 21, 11, 16, 6, 6]
25	146	270	.101	.434	.212	[11, 111, 21, 11, 11, 16, 6]
26	146	270	.101	.434	.212	[11, 111, 21, 11, 11, 11, 16]
27	141	281	.098	.437	.212	[21, 111, 21, 11, 11, 11, 11]
28	146	279	.099	.437	.212	[16, 121, 21, 11, 11, 11, 11]
29	149	281	.098	.438	.212	[16, 131, 21, 11, 11, 11, 11]
30	146	282	.101	.444	.214	[16, 126, 31, 11, 11, 11, 11]
31	146	282	.101	.444	.214	[16, 126, 41, 11, 11, 11, 11]
32	146	282	.101	.444	.214	[16, 126, 36, 21, 11, 11, 11]
33	146	282	.101	.444	.214	[16, 126, 36, 16, 21, 11, 11]
34	146	285	.100	.446	.214	[16, 126, 36, 16, 16, 21, 11]
35	146	285	.100	.446	.214	[16, 126, 36, 16, 16, 16, 21]
36	149	285	.096	.438	.211	[26, 126, 36, 16, 16, 16, 16]
37	141	312	.095	.462	.216	[21, 136, 36, 16, 16, 16, 16]
38	149	285	.096	.438	.211	[21, 131, 46, 16, 16, 16, 16]
39	149	285	.096	.438	.211	[21, 131, 56, 16, 16, 16, 16]
40	149	285	.096	.438	.211	[21, 131, 51, 26, 16, 16, 16]
41	149	285	.096	.438	.211	[21, 131, 51, 21, 26, 16, 16]
42	149	285	.096	.438	.211	[21, 131, 51, 21, 21, 26, 16]
43	149	285	.096	.438	.211	[21, 131, 51, 21, 21, 21, 26]
44	143	312	.095	.462	.216	[31, 131, 51, 21, 21, 21, 21]
45	145	312	.095	.462	.216	[26, 141, 51, 21, 21, 21, 21]
46	143	312	.095	.462	.216	[26, 136, 61, 21, 21, 21, 21]
47	143	312	.095	.462	.216	[26, 136, 56, 31, 21, 21, 21]
48	143	312	.095	.462	.216	[26, 136, 56, 26, 31, 21, 21]
49	143	312	.095	.462	.216	[26, 136, 56, 26, 26, 31, 21]
50	143	312	.095	.462	.216	[26, 136, 56, 26, 26, 26, 31]
51	148	313	.095	.462	.216	[36, 136, 56, 26, 26, 26, 26]
52	149	312	.095	.462	.216	[31, 146, 56, 26, 26, 26, 26]
53	147	337	.093	.480	.218	[31, 156, 56, 26, 26, 26, 26]
54	148	313	.095	.462	.216	[31, 151, 66, 26, 26, 26, 26]
55	148	313	.095	.462	.216	[31, 151, 76, 26, 26, 26, 26]

Table A-3: Clustering results for controller with device instance 3.

run	**ε**	**clusters**	**AMI**	**Hg**	**Cp**	**V_w**
1	19	129	.105	.271	.264	[1, 1, 1, 1, 1, 1, 1]
2	20	129	.093	.260	.255	[11, 1, 1, 1, 1, 1, 1]
3	60	128	.104	.269	.263	[6, 11, 1, 1, 1, 1, 1]

4	77	177	.105	.385	.263	[6, 21, 1, 1, 1, 1, 1]
5	92	173	.107	.384	.263	[6, 31, 1, 1, 1, 1, 1]
6	107	170	.110	.381	.264	[6, 41, 1, 1, 1, 1, 1]
7	**119**	**167**	**.111**	**.377**	**.263**	**[6, 51, 1, 1, 1, 1, 1]**
8	127	162	.109	.363	.260	[6, 61, 1, 1, 1, 1, 1]
9	122	167	.111	.377	.263	[6, 56, 11, 1, 1, 1, 1]
10	122	167	.111	.377	.263	[6, 56, 21, 1, 1, 1, 1]
11	122	167	.111	.377	.263	[6, 56, 16, 11, 1, 1, 1]
12	122	167	.111	.377	.263	[6, 56, 16, 6, 11, 1, 1]
13	122	167	.111	.377	.263	[6, 56, 16, 6, 6, 11, 1]
14	122	167	.111	.377	.263	[6, 56, 16, 6, 6, 6, 11]
15	126	177	.105	.385	.263	[16, 56, 16, 6, 6, 6, 6]
16	138	171	.108	.381	.263	[11, 66, 16, 6, 6, 6, 6]
17	148	168	.111	.378	.264	[11, 76, 16, 6, 6, 6, 6]
18	149	173	.107	.383	.263	[11, 86, 16, 6, 6, 6, 6]
19	147	171	.109	.383	.263	[11, 81, 26, 6, 6, 6, 6]
20	147	171	.109	.383	.263	[11, 81, 36, 6, 6, 6, 6]
21	147	171	.109	.383	.263	[11, 81, 31, 16, 6, 6, 6]
22	147	171	.109	.383	.263	[11, 81, 31, 11, 16, 6, 6]
23	147	171	.109	.383	.263	[11, 81, 31, 11, 11, 16, 6]
24	147	171	.109	.383	.263	[11, 81, 31, 11, 11, 11, 16]
25	149	178	.104	.385	.263	[21, 81, 31, 11, 11, 11, 11]
26	149	194	.092	.399	.258	[16, 91, 31, 11, 11, 11, 11]
27	149	176	.105	.384	.262	[16, 86, 41, 11, 11, 11, 11]
28	149	176	.105	.384	.262	[16, 86, 51, 11, 11, 11, 11]
29	149	176	.105	.384	.262	[16, 86, 46, 21, 11, 11, 11]
30	149	176	.105	.384	.262	[16, 86, 46, 16, 21, 11, 11]
31	149	176	.105	.384	.262	[16, 86, 46, 16, 16, 21, 11]
32	149	176	.105	.384	.262	[16, 86, 46, 16, 16, 16, 21]
33	149	185	.097	.386	.259	[26, 86, 46, 16, 16, 16, 16]
34	107	358	.090	.624	.296	[21, 96, 46, 16, 16, 16, 16]
35	97	363	.089	.630	.297	[21, 91, 56, 16, 16, 16, 16]
36	97	363	.089	.630	.297	[21, 91, 51, 26, 16, 16, 16]
37	97	363	.089	.630	.297	[21, 91, 51, 21, 26, 16, 16]
38	97	363	.089	.630	.297	[21, 91, 51, 21, 21, 26, 16]
39	97	363	.089	.630	.297	[21, 91, 51, 21, 21, 21, 26]
40	100	364	.089	.630	.297	[31, 91, 51, 21, 21, 21, 21]
41	102	364	.089	.630	.297	[26, 101, 51, 21, 21, 21, 21]
42	100	364	.089	.630	.297	[26, 96, 61, 21, 21, 21, 21]
43	100	364	.089	.630	.297	[26, 96, 56, 31, 21, 21, 21]
44	100	364	.089	.630	.297	[26, 96, 56, 26, 31, 21, 21]
45	100	364	.089	.630	.297	[26, 96, 56, 26, 26, 31, 21]
46	100	364	.089	.630	.297	[26, 96, 56, 26, 26, 26, 31]

47	104	365	.089	.631	.297	[36, 96, 56, 26, 26, 26, 26]
48	**103**	**368**	**.088**	**.631**	**.297**	**[46, 96, 56, 26, 26, 26, 26]**
49	110	365	.089	.631	.297	[41, 106, 56, 26, 26, 26, 26]
50	113	364	.089	.630	.297	[41, 116, 56, 26, 26, 26, 26]
51	112	364	.089	.630	.297	[41, 111, 66, 26, 26, 26, 26]
52	112	364	.089	.630	.297	[41, 111, 61, 36, 26, 26, 26]
53	112	364	.089	.630	.297	[41, 111, 61, 31, 36, 26, 26]
54	112	364	.089	.630	.297	[41, 111, 61, 31, 31, 36, 26]
55	112	364	.089	.630	.297	[41, 111, 61, 31, 31, 31, 36]

Table A-4: Clustering results for controller with device instance 4.

run	ε	clusters	AMI	Hg	Cp	V_w
1	18	80	.197	.468	.309	[1, 1, 1, 1, 1, 1, 1]
2	18	95	.182	.504	.305	[11, 1, 1, 1, 1, 1, 1]
3	57	78	.205	.479	.314	[6, 11, 1, 1, 1, 1, 1]
4	78	72	.216	.477	.319	[6, 21, 1, 1, 1, 1, 1]
5	95	72	.216	.477	.319	[6, 31, 1, 1, 1, 1, 1]
6	87	72	.216	.477	.319	[6, 26, 11, 1, 1, 1, 1]
7	87	72	.216	.477	.319	[6, 26, 6, 11, 1, 1, 1]
8	87	72	.216	.477	.319	[6, 26, 6, 6, 11, 1, 1]
9	87	72	.216	.477	.319	[6, 26, 6, 6, 6, 11, 1]
10	87	72	.216	.477	.319	[6, 26, 6, 6, 6, 6, 11]
11	88	78	.205	.479	.314	[16, 26, 6, 6, 6, 6, 6]
12	103	72	.216	.477	.319	[11, 36, 6, 6, 6, 6, 6]
13	116	72	.216	.477	.319	[11, 46, 6, 6, 6, 6, 6]
14	109	72	.216	.477	.319	[11, 41, 16, 6, 6, 6, 6]
15	109	72	.216	.477	.319	[11, 41, 11, 16, 6, 6, 6]
16	109	72	.216	.477	.319	[11, 41, 11, 11, 16, 6, 6]
17	109	73	.217	.481	.320	[11, 41, 11, 11, 11, 16, 6]
18	109	73	.217	.481	.320	[11, 41, 11, 11, 11, 26, 6]
19	**109**	**76**	**.223**	**.519**	**.322**	**[11, 41, 11, 11, 11, 21, 16]**
20	109	76	.223	.519	.322	[11, 41, 11, 11, 11, 21, 26]
21	109	82	.212	.521	.317	[21, 41, 11, 11, 11, 21, 21]
22	122	72	.216	.477	.319	[16, 51, 11, 11, 11, 21, 21]
23	133	72	.216	.477	.319	[16, 61, 11, 11, 11, 21, 21]
24	128	72	.216	.477	.319	[16, 56, 21, 11, 11, 21, 21]
25	128	72	.216	.477	.319	[16, 56, 16, 21, 11, 21, 21]
26	128	72	.216	.477	.319	[16, 56, 16, 16, 21, 21, 21]
27	128	72	.216	.477	.319	[16, 56, 16, 16, 16, 31, 21]
28	128	72	.216	.477	.319	[16, 56, 16, 16, 16, 26, 31]
29	128	75	.211	.477	.317	[26, 56, 16, 16, 16, 26, 26]

30	139	72	.216	.477	.319	[21, 66, 16, 16, 16, 26, 26]
31	149	72	.216	.477	.319	[21, 76, 16, 16, 16, 26, 26]
32	144	72	.216	.477	.319	[21, 71, 26, 16, 16, 26, 26]
33	144	72	.216	.477	.319	[21, 71, 21, 26, 16, 26, 26]
34	144	72	.216	.477	.319	[21, 71, 21, 21, 26, 26, 26]
35	144	72	.216	.477	.319	[21, 71, 21, 21, 21, 36, 26]
36	144	72	.216	.477	.319	[21, 71, 21, 21, 21, 31, 36]
37	144	75	.211	.477	.317	[31, 71, 21, 21, 21, 31, 31]
38	149	89	.213	.548	.322	[26, 81, 21, 21, 21, 31, 31]
39	146	99	.203	.579	.321	[26, 91, 21, 21, 21, 31, 31]
40	149	91	.209	.550	.320	[26, 86, 31, 21, 21, 31, 31]
41	149	91	.209	.550	.320	[26, 86, 41, 21, 21, 31, 31]
42	149	91	.209	.550	.320	[26, 86, 36, 31, 21, 31, 31]
43	149	91	.209	.550	.320	[26, 86, 36, 26, 31, 31, 31]
44	149	91	.209	.550	.320	[26, 86, 36, 26, 26, 41, 31]
45	149	91	.209	.550	.320	[26, 86, 36, 26, 26, 36, 41]
46	149	95	.202	.552	.317	[36, 86, 36, 26, 26, 36, 36]
47	148	104	.191	.580	.313	[31, 96, 36, 26, 26, 36, 36]
48	148	101	.200	.579	.319	[31, 91, 46, 26, 26, 36, 36]
49	148	101	.200	.579	.319	[31, 91, 56, 26, 26, 36, 36]
50	148	101	.200	.579	.319	[31, 91, 51, 36, 26, 36, 36]
51	148	101	.200	.579	.319	[31, 91, 51, 31, 36, 36, 36]
52	148	101	.200	.579	.319	[31, 91, 51, 31, 31, 46, 36]
53	148	101	.200	.579	.319	[31, 91, 51, 31, 31, 41, 46]
54	147	105	.191	.580	.313	[41, 91, 51, 31, 31, 41, 41]
55	**149**	**121**	**.170**	**.615**	**.303**	**[36, 101, 51, 31, 31, 41, 41]**

Table A-5: Clustering results for controller with device instance 5.

run	ε	clusters	AMI	Hg	Cp	V_w
1	19	47	.115	.160	.156	[1, 1, 1, 1, 1, 1, 1]
2	**19**	**57**	**.102**	**.161**	**.149**	**[11, 1, 1, 1, 1, 1, 1]**
3	**61**	**46**	**.117**	**.159**	**.157**	**[6, 11, 1, 1, 1, 1, 1]**
4	83	46	.117	.159	.157	[6, 21, 1, 1, 1, 1, 1]
5	73	46	.117	.159	.157	[6, 16, 11, 1, 1, 1, 1]
6	73	46	.117	.159	.157	[6, 16, 6, 11, 1, 1, 1]
7	73	46	.117	.159	.157	[6, 16, 6, 6, 11, 1, 1]
8	73	46	.117	.159	.157	[6, 16, 6, 6, 6, 11, 1]
9	73	46	.117	.159	.157	[6, 16, 6, 6, 6, 6, 11]
10	76	47	.115	.160	.156	[16, 16, 6, 6, 6, 6, 6]
11	93	46	.117	.159	.157	[11, 26, 6, 6, 6, 6, 6]
12	109	46	.117	.159	.157	[11, 36, 6, 6, 6, 6, 6]
13	101	46	.117	.159	.157	[11, 31, 16, 6, 6, 6, 6]
14	101	46	.117	.159	.157	[11, 31, 11, 16, 6, 6, 6]
15	101	46	.117	.159	.157	[11, 31, 11, 11, 16, 6, 6]
16	101	46	.117	.159	.157	[11, 31, 11, 11, 11, 16, 6]
17	101	46	.117	.159	.157	[11, 31, 11, 11, 11, 11, 16]
18	104	46	.117	.159	.157	[21, 31, 11, 11, 11, 11, 11]
19	116	46	.117	.159	.157	[16, 41, 11, 11, 11, 11, 11]
20	109	46	.117	.159	.157	[16, 36, 21, 11, 11, 11, 11]
21	109	46	.117	.159	.157	[16, 36, 16, 21, 11, 11, 11]
22	109	46	.117	.159	.157	[16, 36, 16, 16, 21, 11, 11]
23	109	46	.117	.159	.157	[16, 36, 16, 16, 16, 21, 11]
24	109	46	.117	.159	.157	[16, 36, 16, 16, 16, 16, 21]
25	112	46	.117	.159	.157	[26, 36, 16, 16, 16, 16, 16]
26	123	46	.117	.159	.157	[21, 46, 16, 16, 16, 16, 16]
27	117	46	.117	.159	.157	[21, 41, 26, 16, 16, 16, 16]
28	117	46	.117	.159	.157	[21, 41, 21, 26, 16, 16, 16]
29	117	46	.117	.159	.157	[21, 41, 21, 21, 26, 16, 16]
30	117	46	.117	.159	.157	[21, 41, 21, 21, 21, 26, 16]
31	117	46	.117	.159	.157	[21, 41, 21, 21, 21, 21, 26]
32	121	46	.117	.159	.157	[31, 41, 21, 21, 21, 21, 21]
33	130	46	.117	.159	.157	[26, 51, 21, 21, 21, 21, 21]
34	125	46	.117	.159	.157	[26, 46, 31, 21, 21, 21, 21]
35	125	46	.117	.159	.157	[26, 46, 26, 31, 21, 21, 21]
36	125	46	.117	.159	.157	[26, 46, 26, 26, 31, 21, 21]
37	125	46	.117	.159	.157	[26, 46, 26, 26, 26, 31, 21]
38	125	46	.117	.159	.157	[26, 46, 26, 26, 26, 26, 31]
39	128	46	.117	.159	.157	[36, 46, 26, 26, 26, 26, 26]
40	137	46	.117	.159	.157	[31, 56, 26, 26, 26, 26, 26]

41	132	46	.117	.159	.157	[31, 51, 36, 26, 26, 26, 26]
42	132	46	.117	.159	.157	[31, 51, 31, 36, 26, 26, 26]
43	132	46	.117	.159	.157	[31, 51, 31, 31, 36, 26, 26]
44	132	46	.117	.159	.157	[31, 51, 31, 31, 31, 36, 26]
45	132	46	.117	.159	.157	[31, 51, 31, 31, 31, 31, 36]
46	135	46	.117	.159	.157	[41, 51, 31, 31, 31, 31, 31]
47	144	46	.117	.159	.157	[36, 61, 31, 31, 31, 31, 31]
48	139	46	.117	.159	.157	[36, 56, 41, 31, 31, 31, 31]
49	139	46	.117	.159	.157	[36, 56, 36, 41, 31, 31, 31]
50	139	46	.117	.159	.157	[36, 56, 36, 36, 41, 31, 31]
51	139	46	.117	.159	.157	[36, 56, 36, 36, 36, 41, 31]
52	139	46	.117	.159	.157	[36, 56, 36, 36, 36, 36, 41]
53	142	46	.117	.159	.157	[46, 56, 36, 36, 36, 36, 36]
54	148	47	.115	.160	.156	[41, 66, 36, 36, 36, 36, 36]
55	145	46	.117	.159	.157	[41, 61, 46, 36, 36, 36, 36]

Appendix B. Evaluation score and cluster amount plots.

Controller 1

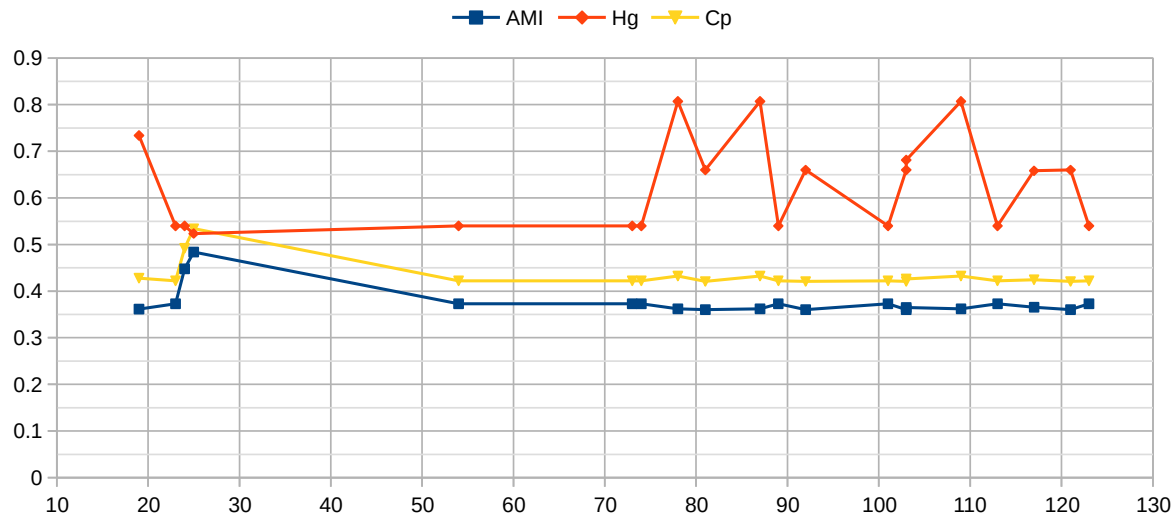

Figure B-1a: AMI, Homogeneity and Completeness against Ɛ.

Figure B-1b: Amount of clusters against Ɛ.

Controller 2

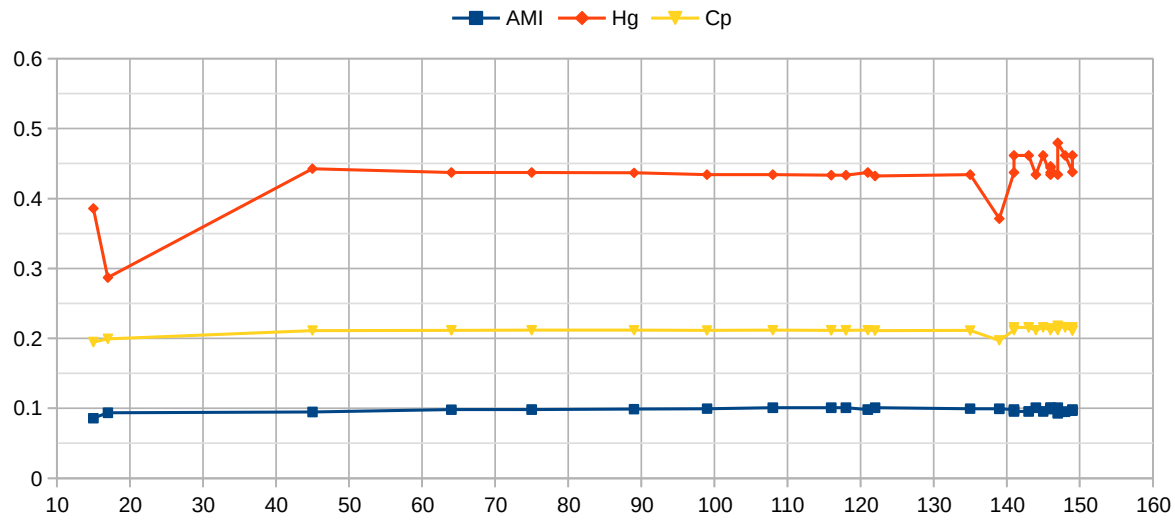

Figure B-2a: AMI, Homogeneity and Completeness against Ɛ.

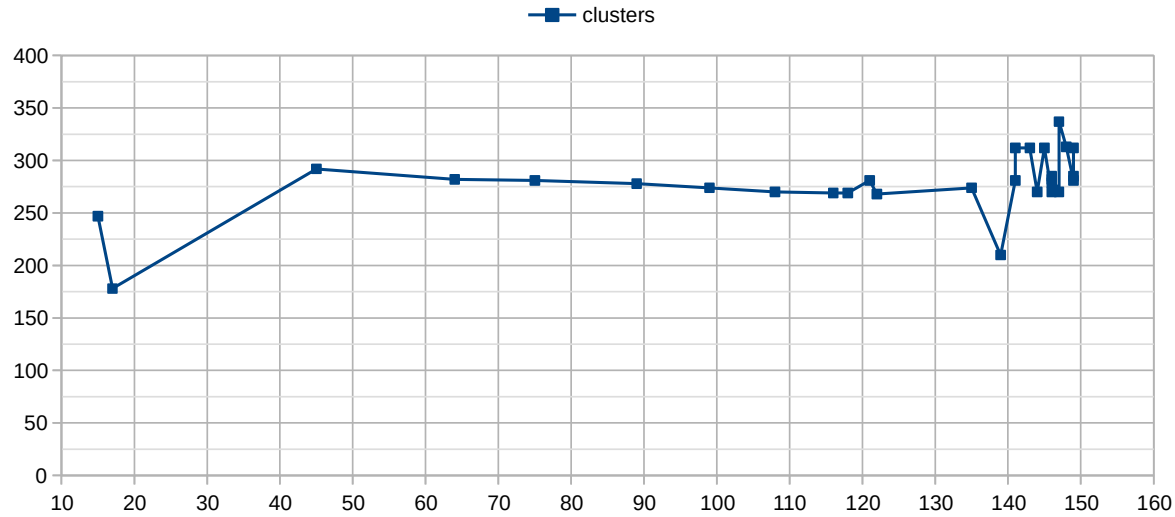

Figure B-2b: Amount of clusters against Ɛ.

Controller 3

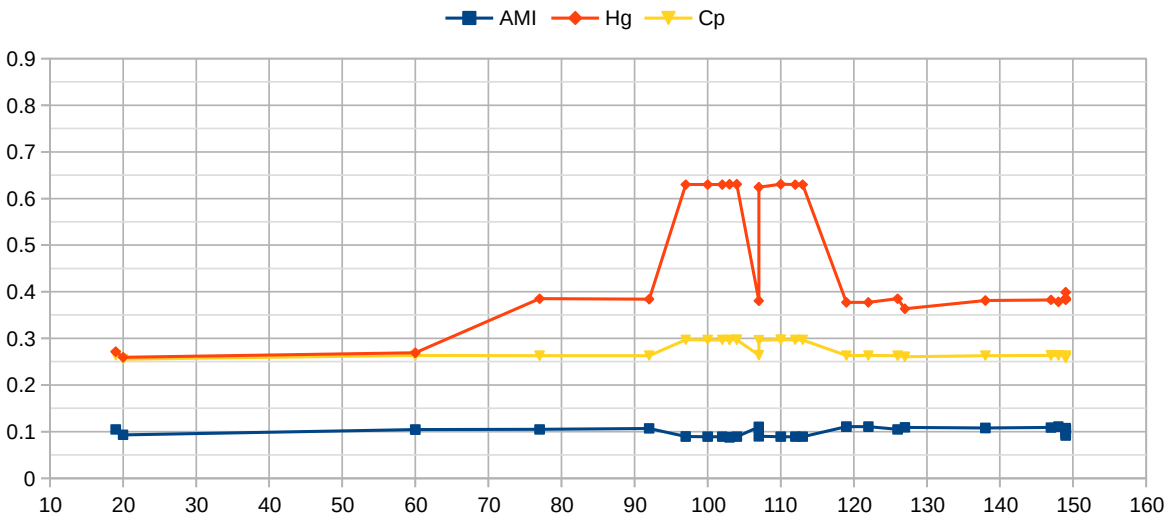

Figure B-3a: AMI, Homogeneity and Completeness against Ɛ.

Figure B-3b: Amount of clusters against Ɛ.

Controller 4

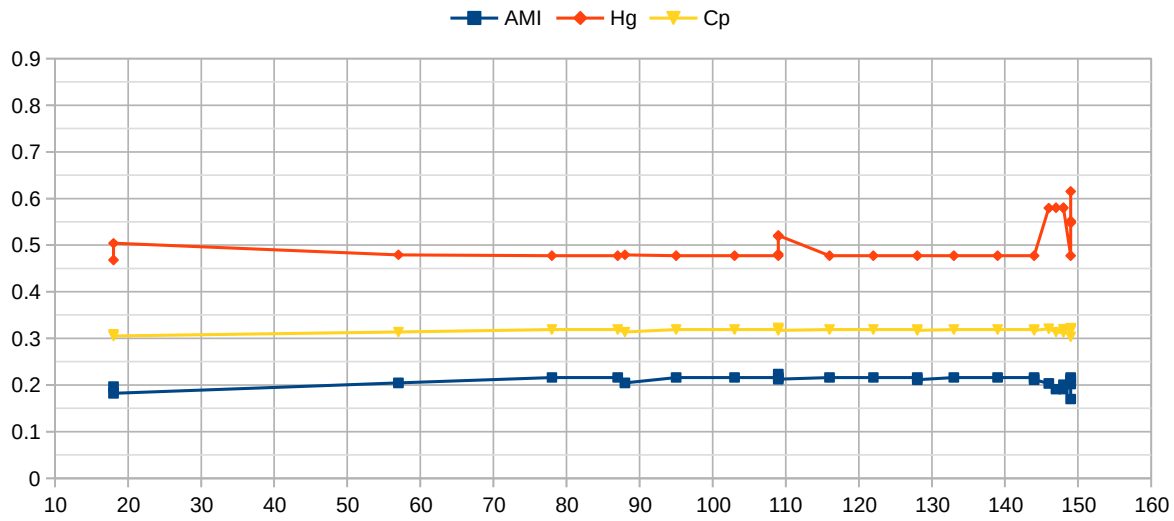

Figure B-4a: AMI, Homogeneity and Completeness against Ɛ.

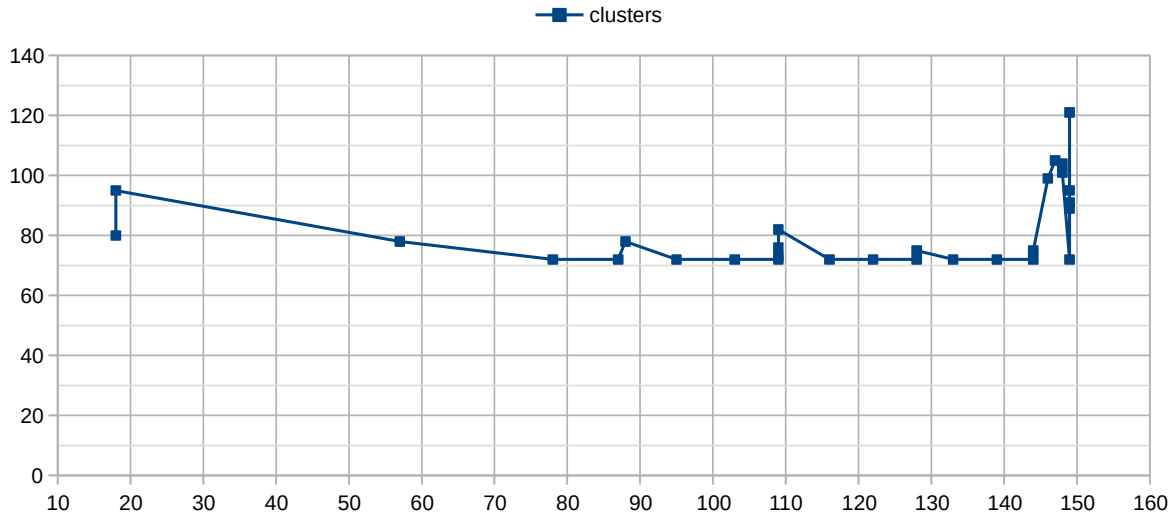

Figure B-4b: Amount of clusters against Ɛ.

Controller 5

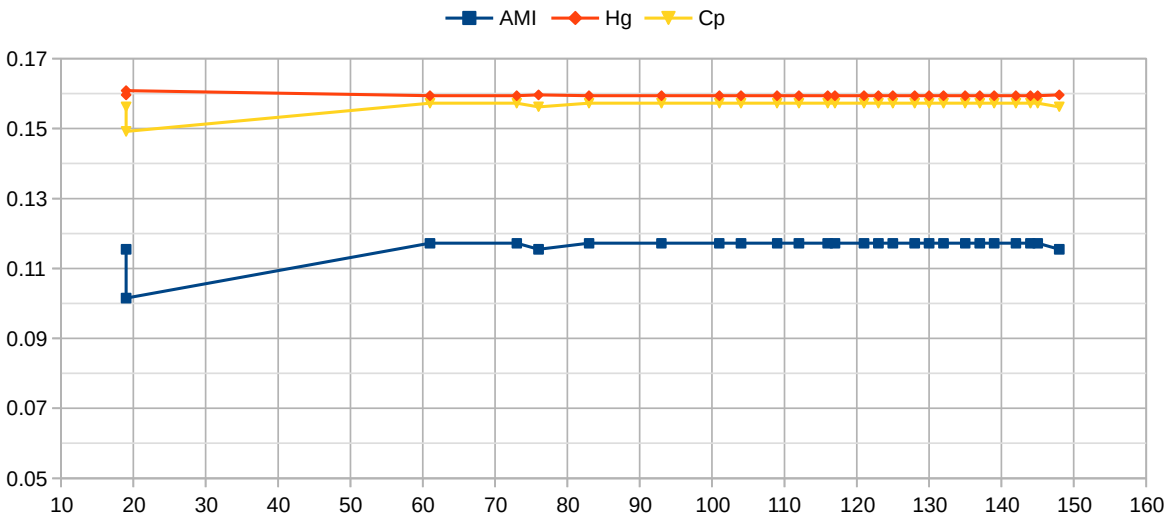

Figure B-5a: AMI, Homogeneity and Completeness against Ɛ.

Figure B-5b: Amount of clusters against Ɛ.