Mälardalen University
School of Innovation Design and Engineering
Västerås, Sweden

Thesis for the Degree of Master of Science in Engineering -
Intelligent Embedded Systems 15.0 credits

MULTI-CONSTRAINED NETWORK OCCUPANCY OPTIMIZATION

Amar Halilovic
ahc19003@student.mdh.se

Nedim Zaimovic
nzc19001@student.mdh.se

Examiner: Radu Dobrin
Mälardalen University, Västerås, Sweden

Supervisor: Tiberiu Secoleanu
Mälardalen University, Västerås, Sweden

May 20, 2020

Abstract

Increasingly different types of information are required in systems, resulting in the diverse and widespread use of networks. The increase in the number of devices on networks significantly increases network traffic and network load, limiting network propagation speed, and increasing the possibility of collisions and lost information. One way to identify the size of the network load is by measuring the network occupancy. Although speeds are constantly increasing, some parts of the network can still become the bottleneck. Therefore, it is always desirable to minimize network occupancy. In this paper, an approach for the multi-constrained network occupancy optimization is presented. The objective is to minimize network occupancy by optimizing the packing process. The problem is identified as a bin packing problem, which is strongly NP-hard. The modification of the Best-Fit Decreasing algorithm is implemented to find the best solution while satisfying multiple constraints. The approach considers grouping signals that are sent to different controllers in the same packages. The analysis is done on the medium-sized plant model, and different topologies were tested. The results show that the proposed solution gives less network occupancy compared to the reference case. Also, by changing the relative placement of network nodes, a significant reduction in network occupancy is achieved.

Table of Contents

1. Introduction	1
1.1. Problem formulation	1
1.2. Solution space	2
1.3. Thesis outline	2
2. Background	4
2.1. Industrial networks	4
2.2. Bin packing problem	6
2.2.1. Mathematical model	7
2.2.2. Heuristic algorithms	7
3. Related Work	9
4. Method	10
4.1. System Development Research Method	10
4.2. Application of the chosen research method to the problem	11
5. Description of the work	12
5.1. System model and initial assumptions	12
5.2. Network package	12
5.3. Network lines/segments	14
5.4. Mathematical model	14
5.5. Reference case	15
5.6. Algorithm	16
5.6.1. Possible improvements	16
5.6.2. Our approach	17
5.7. Architecture	18
5.7.1. Network and devices	19
5.7.2. Package, Signal, Task, Dataset	19
6. Results	21
6.1. Implementation details	21
6.2. Topology 1	22
6.2.1. Reference case	22
6.2.2. Proposed solution	23
6.2.3. Comparison	24
6.3. Topology 2	25
6.3.1. Reference case	26
6.3.2. Proposed solution	26
6.3.3. Comparison	27
6.4. Topology 3	28
6.4.1. Reference case	28
6.4.2. Proposed solution	29
6.4.3. Comparison	30
7. Discussion	32
7.1. Influence of different packing on network occupancy	32
7.2. Influence of the BFD algorithm on package utilization	32
7.3. Influence of topology on network occupancy	34
8. Conclusions	35
8.1. Future work	35
References	38

List of Figures

1	Hierarchy of an industrial automation system [3]	1
2	Controller-Centric architecture	4
3	Network-Centric architecture	5
4	Ethernet IEEE 802.3 frame [15]	5
5	Bin packing problem	6
6	<i>System Development Research Method</i> phases and interconnections between them [27]	10
7	Steps in the <i>System Development Research Method</i> [27]	11
8	Network-Centric Control Model	12
9	OPC UA message layers [28]	13
10	UADP dataset payload [28]	13
11	Package structure	13
12	Simple network	14
13	Reference case packing procedure	16
14	Illustrative representation of the first phase of the algorithm	18
15	Network model	19
16	Package, Signal, Task and Dataset classes	20
17	Sending schedule of packages and their content	21
18	Topology 1	22
19	Topology 2	25
20	Topology 3	28

List of Tables

1	Upper bounds	8
2	Example of dataset assignment in FCI 1	15
3	Algorithm parameters	21
4	Network occupancy for the reference case (Topology 1, $H = 33B$, $T = 9s$)	23
5	Network occupancy for the proposed solution without allowed misses (Topology 1, $H = 33B$, $T = 9s$)	23
6	Network occupancy for the proposed solution with allowed misses (Topology 1, $H = 33B$, $T = 9s$)	24
7	Network occupancy reduction without allowed misses (Topology 1, $H = 33B$, $T = 9s$)	24
8	Network occupancy reduction with allowed misses (Topology 1, $H = 33B$, $T = 9s$)	25
9	Network occupancy for the reference case (Topology 2, $H = 33B$, $T = 9s$)	26
10	Network occupancy for the proposed solution without allowed misses (Topology 2, $H = 33B$, $T = 9s$)	26
11	Network occupancy for the proposed solution with allowed misses (Topology 2, $H = 33B$, $T = 9s$)	27
12	Network occupancy reduction without allowed misses (Topology 2, $H = 33B$, $T = 9s$)	27
13	Network occupancy reduction with allowed misses (Topology 2, $H = 33B$, $T = 9s$)	28
14	Network occupancy for the reference case (Topology 3, $H = 33B$, $T = 9s$)	29
15	Network occupancy for the proposed solution without allowed misses (Topology 3, $H = 33B$, $T = 9s$)	29
16	Network occupancy for the proposed solution with allowed misses (Topology 3, $H = 33B$, $T = 9s$)	30
17	Network occupancy reduction without allowed misses (Topology 3, $H = 33B$, $T = 9s$)	30
18	Network occupancy reduction with allowed misses (Topology 3, $H = 33B$, $T = 9s$)	31
19	Difference between reference and proposed solution in number of packets	32
20	Difference between reference and proposed solution in the packets utilization	32
21	Allowed misses and size of signals in dataset S1-S20	33
22	Part of FCI1 packing schedule without allowed misses	33
23	Part of FCI1 packing schedule with allowed misses	33
24	Difference between topology 1 and topology 2 in occupancy of segments 4, 5, and 10	34
25	Difference between topology 2 and topology 3 in occupancy of segments 4,5, and 10	34

List of Abbreviations

BF	Best-Fit
BFD	Best-Fit Decreasing
FCI	Field Control Interface
FF	First-Fit
FFD	First-Fit Decreasing
IoT	Internet of Things
JMS	Job Management System
LCM	Least Common Multiple
MAC	Media Access Control
NkF	Next-k-Fit
OOP	Object-Oriented Programming
OPC	Open Platform Communication
OPC UA	OPC Unified Architecture
SFD	Start Frame Delimiter
UADP	Unified Access Data Plane
WF	Worst-Fit

1. Introduction

Increasingly different types of information are required in systems, resulting in the diverse and widespread use of networks. Today, more and more devices are connected to networks. Estimates are that there are currently 25 Billion Internet of Things (IoT) devices deployed in the world, and that number has increased for about 20 Billion in the last five years [1]. Therefore, we can expect communications-capable machines to be the most common type of device in the future. Simultaneously, the volume of the data will also grow. The increase in the number of devices on networks significantly increases network traffic and network load, which limits and even decreases the network propagation speed and increases the possibility of lost information in the network. Powerful data communication is necessary to handle this growth.

Network occupancy plays an important role. It shows how big is the network load. Although network speeds are getting higher, it is always desirable to minimize the network occupancy. Because of this, network optimization plays a vital role in many systems and applications that have certain speed and reliability requirements. These requirements mean that it is required to deliver packages accurately to their destination in some guaranteed time frame. Multi-constrained optimization is one of the most common real-world engineering problems. It is challenging to find and guarantee an optimal solution. That is why we use algorithms that find sub-optimal solutions that are satisfactory under the given conditions, subject to the given constraints. Among the most well-known algorithms is a group of heuristic algorithms, which are guided by informal mechanisms such as knowledge, intuition, and experience. Heuristics are sometimes used instead of classical approaches when classical approaches do not yield satisfactory results.

1.1. Problem formulation

The context of our work here is triggered by a move from the classical *Control-Centric System Model* towards a *Network-Centric Control Model*. It is concerned with the acquisition, delivery, and processing of input signals from sensors to controllers and delivery of the controller signals to actuators. Such an industrial distributed control system is deployed on an Ethernet network technology, over which the Open Platform Communications Unified Architecture (OPC UA) [2] protocol is running. We will focus on the part of the model between sensors and controllers - see Figure 1.

Figure 1: Hierarchy of an industrial automation system [3]

Sensors are placed at the bottom of an industrial automation system. They provide the necessary data to the control system, such as the presence of an object or a physical property, or a temperature, etc. Signals from sensors are aggregated in Field Control Interfaces (FCI) to reduce the network load and grouped in larger datasets. Every FCI has several signal datasets in the network which are mutually disjunctive to datasets in other FCIs. On the other part of the network are controllers which are assigned tasks. Each task is assigned a dataset of signals to process. These datasets overlap with datasets of signals in the FCIs. FCIs have registers storing signal values at each moment. Each dataset in the FCI has an *ideal* frequency by which the FCI assembles signals from the dataset and packs them into packets.

The goal is to find the optimal packing schedule, maximize the packet size utilization, and minimize the number of packets sent so that the network occupancy, expressed in Bytes (B), is minimized. All these, with minimal changes in the ideal frequency of the datasets involved.

The objective of this research is to answer the following research questions:

RQ1: *How can the network occupancy be minimized between the sensors and controllers in a network-centric control model?*

RQ2: *How does the different packing of input data affect the network occupancy?*

RQ3: *How does the choice of the network topology affect the network occupancy?*

Contribution. Based on the problem formulation and formulated research questions, we contribute to the problem solution in the following ways:

- Identify and adapt the suitable algorithm that minimizes network occupancy given the multiple constraints.
- Determine how different packing of the data affects the network occupancy in a network-centric model.
- Determine how the choice of the network topology affects the network occupancy in a network-centric model.

1.2. Solution space

The first direction of study brings us to the rich body of research related to “bin packing” [4, 5, 6, 7]. There are many variants and forms of “bin packing” problem and solutions to it. However, each of them tries to fit the finite number of items into the finite number of bins of a fixed volume so that the number of bins is minimal. This problem fits in the class of the 1D packing, where only one dimension varies, and that is the size of the signal. In this problem, packets represent bins of a fixed maximum size, while signals from sensors represent the items which size varies.

Heuristic and metaheuristic algorithms and approaches [8] are generally used for finding sub-optimal solutions or solutions that are good enough under given conditions when classical approaches are too slow or fail to find optimal solutions. Such approaches exist that speed-up “bin packing” solutions [9, 10].

Another direction would be to investigate the use of various artificial intelligence algorithms (e.g., genetic algorithms, as in [11]), or more sophisticated approaches. These approaches can be applied to a larger class of problems and utilized if some more problem-specific algorithms do not yield satisfactory results. The shortcomings of these types of algorithms are that they are typically more complex and require adjustment to fit the needs of the specific problem.

For the size of the problem here, we pursue the bin packing approach initially, remaining to look into further solution dimensions if the results are not convincing, or not sufficiently useful.

1.3. Thesis outline

The report contains eight sections. The current, first section, is the introductory section in which we formulate the problem, and present possible paths to the solution, after which in section 2, we present the theoretical background and mathematical apparatus. In section 3, we present a review

of related work. Further, in section 4, the chosen research methodology, as well as how it applies to the problem, is described. Section 5 contains a system model and a mathematical model, as well as a description of the developed algorithm. Section 6 provides results, together with simulation settings. A discussion on the results is taken in section 7, after which we conclude, while also introducing some possible ideas for improvement through future work, in section 8.

2. Background

2.1. Industrial networks

Networks allow us to make various computing devices interact with each other over a shared medium, which enables the transfer of vast amounts of the data between devices over long distances and large spaces. End devices that originate and terminate data connection together with devices that forward and route data are known as network nodes. Transmitted data over a network are generally packed into smaller chunks called packets. Packets consist of control data known as packet header and user data known as the payload. Control data, among other things, tells nodes to which destination to send the packet, while payload contains actual data to be read by the receiver node. Arrangement of all nodes in one network is a network topology, which can describe various arrangements of network nodes and types of networks [12], including the industrial networks.

Industrial plants are almost always very complex systems with a large number of computing devices. Some of the most common industrial devices are sensors, controllers, and actuators, which are the main components of control systems. The main target is connecting all plant sections into a single network infrastructure to make industrial communication work from the field level up to the management level [13].

The switching technology is one among many technologies used in modern industrial networks. It uses devices called hubs, and network switches. When a packet enters a switch, the switch reads the Medium Access Control (MAC) address stored in the packet header. Based on address value, it makes decision through which of its ports to send the packet, while the hub forwards the packet to all its ports. While doing this, it uses the store-and-forward transmission principle [14]. This mechanism decreases the packets' collision probability and thus enhances the network speed.

Process control and automation are very demanding and require scalable and flexible systems. Many technological advancements are made every year to satisfy these requirements. One technology that significantly contributed to the advancement of the industrial networks is the Ethernet technology.

Controller-Centric Systems. Figure 2 depicts a logical topology of the controller-centric architecture.

Figure 2: Controller-Centric architecture

Devices in the system are directly connected to controllers, thus making controllers to “own” devices. Configuration data from the engineering to devices is deployed over controllers. Controllers require certain knowledge about device-specific implementations and are not only focused on control logic execution. Routing of device information goes through the controllers.

Network-Centric Systems. Figure 3 depicts a logical topology of the network-centric architecture.

Figure 3: Network-Centric architecture

Network-centric architecture brings many benefits compared to the control-centric architecture. As devices and controllers are logically on the same bus, any controller can use signals from any device. The engineering of devices and controllers can be done independently and separately. This approach relieves the controllers of the source signals and provides system-wide easy access to device information. Overall, the system is more flexible. However, this also brings some drawbacks, one of which is the possible network congestion in large systems.

The Ethernet. Many common network types use Ethernet technology such as local area, metropolitan area, and wide area networks. Among many things, Ethernet defines medium types, network speed, and packet design. Ethernet has evolved through its history through standards. Every Ethernet standard predefined existing and defined new features supported by this technology. As defined in the newer standards, Ethernet now supports speeds in the Gigabit domain and both fiber optic and twister-pair cabling [15].

In Ethernet standard IEEE 802.3, a packet is called a frame, even though the frame is actually only a part of the packet. The packet consists of the preamble, start frame delimiter (SFD), carrier extension, and frame. The frame is usually a part after carrier extension and consists of a header, payload, and frame check sequence. Frame header includes destination and source MAC address and payload length. [15].

Figure 4 depicts the Ethernet IEEE 802.3 frame.

Figure 4: Ethernet IEEE 802.3 frame [15]

The development of Ethernet technology has produced devices that have significantly improved automation and network infrastructure and have widened data access capabilities.

OPC. OPC stands for Open Platform Communication [16], which is a set of standards that defines the secure interoperability of data in industrial automation and telecommunication. The standard was first developed for process control systems, but now it spans beyond process control to industries like manufacturing and oil and gas. It allows communication between devices from different manufacturers and vendors and relatively cheap integration of different platforms without

a time-consuming development process. OPC has expanded over the years into a multi-billion-dollar industry, and there are now more than 4000 vendors that supply more than 35000 different OPC products used in over 17 million applications [17]. To address new security and modeling challenges that have arisen with the development of service-oriented architectures, OPC Unified Architecture (OPC UA) is introduced. It is an extensible and scalable open-platform architecture, rich in features. One of the features is the communication model, which describes a publish-subscribe pattern. Publishers and subscribers are loosely coupled entities that send and receive messages, respectively. They often will not even know each other. Their primary relationship is a shared understanding of specific types of data, which are used for communication. OPC is continuously developed and is now supporting a wide range of technologies.

2.2. Bin packing problem

Suppose we have a set of items, each of a given volume (usually different), and some bins or containers each of a fixed volume (usually same). The goal is to pack the items to the bins in a way that minimizes the number of bins used (Figure 5).

Figure 5: Bin packing problem

The total size of the items assigned to one bin should not exceed its capacity. This problem is known as the bin packing problem, and it is a very popular challenge amongst the researchers across various domains. There are many variants of this problem, such as three-dimensional packing [18], packing by weight [19], packing by cost [20], etc. They have many applications, such as filling up containers [21], loading trucks with weight capacity constraints [22], and job scheduling [23]. The bin packing problem is strongly NP-hard problem, and it is proven that no algorithm can achieve a performance ratio better than $\frac{3}{2}$ (unless $P = NP$) [24]. Heuristic methods are most commonly used to find the optimal solution to this problem. The heuristic algorithms can be divided into two categories:

- Online algorithms
- Offline algorithms

In the online algorithms, items that need to be packed arrive one at a time in an unknown order. Each of them must be put in a bin before considering the next item. In the offline version, we have all items and their sizes available before the process of packing.

2.2.1. Mathematical model

The mathematical definition of the classical single capacity bin packing problem can be formulated as follows.

Let $\mathbf{L} = \{x_1, x_2, \dots, x_n\}$ be the set of items that need to be packed. Each item x_i has a given size s_i , such that $0 < s_i \leq C$, where C is the capacity of the bin.

$$y_i = \begin{cases} 1 & \text{if bin } i \text{ is used} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$x_{ij} = \begin{cases} 1 & \text{if item } j \text{ is assigned to bin } i \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The objective is to

$$\text{minimize } k = \sum_{i=1}^m y_i \quad (3)$$

subject to

$$k \geq 1 \quad (4)$$

$$\sum_{i=1}^m x_{ij} = 1 \text{ for } j \in \{1, \dots, n\} \quad (5)$$

$$\sum_{j=1}^n s_j x_{ij} \leq C y_i \text{ for } i \in \{1, \dots, m\} \quad (6)$$

Equation 3 represents the goal of optimization, which is to minimize the number of packages. Equation 5 limits the presence of one item to only one container, while Equation 6 limits the total size of the stored items.

2.2.2. Heuristic algorithms

One of the most simple heuristics for bin packing is probably Next-Fit (NF). NF places an item in the currently "opened" bin if the item fits inside. Otherwise, the current bin is closed, and the item is placed inside the new bin. The NF modification is a Next-k-Fit (NkF) which, unlike the NF, holds k containers open instead of only one. First-Fit (FF) is one of the most commonly used heuristic algorithms for solving the bin packing problem. The algorithm works as follows. Each item in $\mathbf{L}$ is placed into the first bin in which it will fit. If there is no such bin, it puts the item inside the new bin. Another well-known algorithm that provides a fast but often non-optimal solution is Best-Fit (BF). The idea is to put the item in the fullest bin in which it fits. Worst-Fit (WF) is very similar to BF. Instead of putting an item in the bin where it leaves the smallest space, it is placed in the bin with minimum load. Since all these algorithms belong to the group of online algorithms, they consider items in an order defined by a list $\mathbf{L}$.

To measure the performance of the algorithm, two terms are defined:

- $A(\mathbf{L})$ - The number of bins used when algorithm A is applied to list $\mathbf{L}$
- $OPT(\mathbf{L})$ - The optimum number of bins for list $\mathbf{L}$

Table 2 shows the upper bounds for each of the mentioned algorithms.

Algorithm	Upper bound
Next-Fit	$NF(\mathbf{L}) \leq 2OPT(\mathbf{L}) - 1$ [24]
First-Fit	$FF(\mathbf{L}) \leq \lfloor 1.7OPT(\mathbf{L}) \rfloor$ [6]
Best-Fit	$BF(\mathbf{L}) \leq \lfloor 1.7OPT(\mathbf{L}) \rfloor$ [7]
Worst-Fit	$WF(\mathbf{L}) \leq 2OPT(\mathbf{L}) - 1$ [24]

Table 1: Upper bounds

Since FF and BF have lower upper bounds than NF and WF, the two most commonly used offline algorithms are Best-Fit Decreasing (BFD) and First-Fit Decreasing (FFD). Both algorithms initially sort the items in decreasing order of size and assign the larger items first. In this way a lower upper bound is achieved [25]:

$$FFD(\mathbf{L}) \leq \frac{11}{9}OPT(\mathbf{L}) + \frac{6}{9} \quad (7)$$

3. Related Work

Leinberger et al. [4] proposed a new multi-capacity aware bin packing algorithm for job management systems (JMS). Multi-capacity refers to different resource requirements, such as the number of CPUs, amount of memory, etc. The previous JMS have led some resources to become exhausted while others remain under-utilized. The contribution of the work is to use heuristics to select items that attempt to correct a capacity imbalance in the current bin. The bin represents the parallel system, while the job wait queue is represented by an item list where a d-capacity requirements vector describes each item. The algorithm searches the list looking for an item that fits in the bin and in which specific set of resource requirements is largest. This set of requirements is chosen to lessen the capacity imbalance.

In the paper presented by Sheng et al. [21], the container loading problem with expiring orders is addressed. The authors classified this problem as a three-dimensional, container-loading problem with multiple constraints such as orientation, stability, and the loading priority constraint. The items of an order must be entirely placed in the container or must be entirely left behind. A heuristic algorithm is proposed to solve this problem, which first handles the expiring and then the non-expiring orders. The algorithm works in two steps. In the first, a simulated annealing strategy is used to select a collection of orders, and in the second, a tree search procedure is used to pack them into the container. Although several constraints were considered, some constraints had not been taken into account, such as weight.

In [5], Omar presented a comparison between the most commonly used heuristic algorithms for solving bin packing problems. The optimization problem was multi-objective: minimize the bins used and minimize the heterogeneousness of the elements in each bin. The authors emphasized the importance of trade-off in this kind of optimization problem. There is no single solution in the space of feasible solutions that equally minimize the two objective functions. Therefore, these two conflicting goals were formulated as a vector optimization problem. The bin packing was considered as a two-tier problem: reordering candidate objects and packing them. The following orders were tested: as given, regular, random, ascending, and descending. Out of known packing algorithms, First-Fit, Next-Fit, Random-Fit, and Best-Fit were combined with different orders and applied to the given problem. Best-Fit with descending order of items turned out to be the best, but still, it did not provide the best solution.

Bin packing problem under linear constraints is presented by Wang et al. [26]. Unlike the classic bin packing problem, the size of items to be packed is not given in advance. The size of items is also part of the decision and must satisfy a set of linear constraints. Since all constraints are known, the algorithm is in the offline setting. The authors proved that this is an NP-hard problem in a strong sense. Modified Next-Fit algorithm is proposed, which is a combination of linear programming and classic Next-Fit. The linear program is used to determine the size of items, and then, Next-Fit algorithm is deployed to solve the bin packing problem. The algorithm runs in polynomial time and has the same approximation ratio as classic Next-Fit. There is a space left to modify other known heuristic bin packing algorithms in this way and maybe achieve a better performance ratio.

In [11], Salimi et al. presented a meta-heuristic approach for real-time task scheduling problem for distributed processing environments. It is a multi-population evolutionary approach that guarantees end-to-end tasks' deadlines in distributed environments. The goal was achieving a near-optimal scheduling optimization, which is expected considering the meta-heuristic approach. They analyzed two scenarios: one with single-objective, another with multi-objective optimization. In a single-objective optimization scenario, they tried to minimize the number of processing units for all tasks. In contrast, in a multi-objective optimization scenario, they tried to optimize the total number of processing units and end-to-end delay of all jobs, which represent two conflicting objectives. The experiments were based on the industrial automation systems on the basis of which the synthetic use case was derived. Their approach proved to offer better resource efficiency and better optimization in both scenarios than similar evolutionary approaches while guaranteeing real-time performances. The algorithm was not tested on more complex problems, which include task duplication and synchronization activities, which can impact reliability and fault tolerance.

4. Method

4.1. System Development Research Method

The main goal of the thesis is to develop a new approach for a multiple-constrained network occupancy optimization. Having this in mind, the suitable research method for this thesis is *System Development Research Method*, first presented by Nunamaker and Chen [27]. It comprises four phases, which are all mutually interconnected:

- Theory Building
- System Development
- Experimentation
- Observation

Figure 6 depicts these four phases and their interconnections.

Figure 6: *System Development Research Method* phases and interconnections between them [27]

The first phase is the *Theory Building* phase in which the conceptual frameworks, mathematical models, and methods are investigated and new ones built. This phase serves as a theoretical background for next phases. Then comes the *System Development* phase which includes prototyping and constructing the system architecture, product development and possibly technology transfer. The third phase is the *Experimentation* phase, in which one or more experiments or simulations are performed to test implemented theoretical ideas in the second phase. Experiments can be lab or field-based, depending on the context of the research. Computer simulations are often used because they are generally cheaper to perform than experiments. The *Observation* phase helps researchers understand the scope of the research and all things involved in it by using case, survey and field studies.

As every phase interconnects with all others, it is feasible to jump between phases and manipulate the data from other phases. This flexibility improves the development process.

4.2. Application of the chosen research method to the problem

The research procedure employed in this thesis is based on the aforementioned *System Development Research Method*. As this method is a multi-methodological research approach, it consists of multiple steps, which are shown in figure 7.

Figure 7: Steps in the *System Development Research Method* [27]

These steps are modified to fit this thesis research needs best, and the whole research process is summarized in the next steps:

1. Analysis of a system model, its components, and requirements.
2. Analysis of the existing optimization approaches and techniques that can be applied to the model.
3. Selection of the most suitable optimization technique(s).
4. Design of the new optimization technique using the elements of the selected existing optimization technique(s).
5. Implementation of the new optimization technique.
6. Simulation, evaluation of the results, and report writing.

As this method is an iterative process, every step of the research process can be refined based on the output of any other step. A model of the network and network devices is appropriately defined to validate obtained results so that they hold to the real object, with an adequate number of variables so the optimization algorithm can utilize them.

5. Description of the work

5.1. System model and initial assumptions

The *Network-Centric Control Model* models the acquisition and delivery of the input signals from sensors to controllers, processing the signals in the controllers and delivery of the controller signals to actuators on the *Smart Control Platform* on the plant. The model represents the industrial distributed control system with an Ethernet network technology deployed, over which an OPC UA protocol is running. Figure 8 depicts the network-centric control model.

Figure 8: Network-Centric Control Model

Our focus lies on the part of the model between sensors and controllers. Sensors are modeled as device nodes and directly connected to the network bus, or the devices called Field Controller Interfaces (FCIs). In this work, we will assume that sensors can only be connected directly to the FCIs. Each FCI is assigned a certain number of sensors. Signals from assigned sensors are aggregated in the FCI in larger datasets to reduce the network load. Aggregated signals in one FCI are divided into datasets that are mutually disjunctive to datasets in all other FCIs in the network. The signal from each sensor is stored in the corresponding register in the FCI. The time period in which the signal sends the data is negligible, and it is assumed that FCI has values from every signal assigned to it at every moment in its registers.

The network is modeled as an ideal network without delays and package losses. The network occupancy is a parameter defined as a number of bytes that flows through the network over a specified time interval. In the network, there is a number of switches that forwards packets to the right controllers. Each switch is directly connected to at least one controller, FCI, and another switch. Depending on the choice of the network topology, the number of network elements, their connections, and their disposition varies. On the other part of the network are controllers which are assigned tasks. Each task is defined by its period and execution time and is assigned a dataset of signals which it processes. These datasets overlap with datasets of signals in the FCIs.

5.2. Network package

Sometimes, the term messages can have multiple meanings in networks. To avoid confusion, in this paper we will consider payload as a message. So the data read from the sensor is a payload that is sent to the network together with the rest of the package. The network package also includes

protocol, security, and encoding-specific data. Within the FCI, the entire package is created and as such sent to the appropriate controller. All message layers are shown in Figure 9.

Figure 9: OPC UA message layers [28]

The DataSetMessage is actually data from a single sensor, while NetworkMessage is the overall payload that contains all signals. The Unified Access Data Plane (UADP) dataset payload and other parts of the network package are shown in Figure 10.

Figure 10: UADP dataset payload [28]

We can see that the package in addition to payload consists of different types of headers, security footer and signature. Everything except payload can be considered a fixed size. In addition to messages, the payload also contains their sizes. For simplicity, we will consider that this information is contained in the message itself. A simple representation of the package is shown in Figure 11.

Figure 11: Package structure

The messages contained in the payload are sensor signals. The data size of the signal is marked as DS, and it depends on the type of signal. The signal can be:

- Analog signal - 15B
- Digital signal - 12B
- Structured signal - 105B

5.3. Network lines/segments

In Figure 12, one simple network topology is shown, and below we will use it to present the algorithm. Each controller-controller, switch-switch, and the switch-controller connection is marked with a number. These numbers are used to represent network lines/segments. Any information sent to the network goes through some of these segments to its destination. As we consider only one direction of data transfer, the package's final destination is in one of the controllers. Even with the example of the relatively small network shown in Figure 12, we can see that the network lines occupancy depends a lot on the relative position of the elements in the network.

Figure 12: Simple network

For example, let assume that FCI 1 sends data to controller 1 and controller 4. When FCI 1 sends data to controller 4, packages pass through segments 3, 4, and 5. On the other hand, when it sends a package to controller 1, the package only passes through segment 1. If any of the FCIs attached to switch 2 (SW 2) or switch 3 (SW 3) wants to send data to controller 1 or controller 2, it would further occupy segment 3. It is expected that the segments connecting the two switches will be the most loaded with data. What further affects the occupancy of the segments is the frequency of packages being sent. The higher the frequency, the higher the network occupancy.

5.4. Mathematical model

The size of the package is defined with the following equation (Eq. 8):

$$p_i = H + \sum_{k=1}^s x_{ik} DS_k \quad (8)$$

$$p_i \leq 1500 \quad (9)$$

$$\sum_{i=1}^n x_{ik} = 1 \quad (10)$$

$$x_{ik} = \begin{cases} 1 & \text{if signal } k \text{ is put into package } i \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

where

- p_i - size of package i (bytes)
- H - Size of header and other constant (non-data) parts of the package
- DS_k - Data size of signal k
- s - The number of different signals inside the package

- n - The number of packages

Equation 9 means that the total size of the loaded signals is not greater than the size of the package. Equation 10 defines that each signal is only included in one package.

As mentioned before, the goal is to reduce network occupancy as much as possible. We will try to optimize network occupancy at a specific time interval. It is crucial to emphasize how network occupancy is calculated. Equations 12 and 13 define segment occupancy as well as total network occupancy.

$$O_i = \sum_{j=1}^n \left\lceil \frac{T}{T_j} \right\rceil p_j \quad (12)$$

$$O_{tot} = \sum_{i=1}^s O_i \quad (13)$$

where

- O_{tot} - Total network occupancy
- O_i - Occupancy of the segment i
- T - The time over which the network occupancy is observed
- p_j - Size of the package j
- T_j - Reading (packing) period of the package j
- $\left\lceil \frac{T}{T_j} \right\rceil$ - The maximum number of times package j is sent within a specified time interval
- n - The number of different packages that are sent through segment i
- s - The number of segments

5.5. Reference case

Table 2 shows one possible scenario for the distribution of datasets to controllers within FCI 1.

Controller	Data Set	Ideal reading period
CTL3	S1-S20	50
CTL4	S15-S40	100
CTL5	S41-S70	150

Table 2: Example of dataset assignment in FCI 1

We can see that each dataset has an ideal reading period, i.e., the period during which this data should be packed and sent to the specific controller. Often, all the data intended for one controller cannot fit into one package. Therefore, the data are sent in multiple packets. The mechanism that packs the signals into packets goes in sequence through the dataset and packs the signals one by one. If the signal does not fit into the package, a new one is opened, and the packing procedure is continued. This packaging procedure is the same as the packing procedure of the NF algorithm. Figure 13 shows one possible appearance of packages sent from FCI 1 to CTL 1.

Figure 13: Reference case packing procedure

In this way of packing, we can see that some packages are not being used to the maximum. This can lead to unnecessary creation of new packages even though the data may fit into fewer packages. Further, let consider a time instant of 100ms when datasets S1-S20 and S15-S40 are both collected and sent to the appropriate controllers. Observing the topology in Figure 12, we can see that these datasets will travel through segments 3 and 4 at the same time. As the datasets are placed in different packages, this will result in an increase in the occupancy of segments 3 and 4. Data may be sent together in the same package to reduce network occupancy. The drawback of such an approach is that controller 4 would, in addition to the requested data, also receive data intended for controller 3. Therefore, the occupancy of segment 5 will increase. This example demonstrates the importance of trade-off in this type of optimization problem. Using our algorithm, we will try to find a solution that minimizes the number of packages and thus the data in the network.

5.6. Algorithm

The goal of the algorithm is to minimize the number of packets in the network, i.e., minimize the network occupancy. In the previous subsection, we highlighted the shortcomings of the current packing method. We also presented some ideas on how the existing algorithm could be improved. This subsection shows the implementation of the algorithm.

5.6.1. Possible improvements

Often signals from the same FCI are sent to different controllers at the same time. As a result, some network lines become occupied by a large number of packets. Given that there are dozens of such FCIs in the network, the load on these segments becomes enormous. One way to influence network occupancy is to consider merging data from different datasets into the same package. If data intended for different controllers share most of the way through the network, putting them together instead of in separate packets can reduce network occupancy. The disadvantage of this approach is that controllers would receive additional data that they would have to discard in some way. Also the occupation of smaller branches in the network would increase. This is a trade-off that must be taken into account when solving this issue.

As mentioned earlier, the packing algorithm in the reference case leaves the packages unfilled. This can lead to unnecessary splitting of the dataset into multiple packages. When a large amount of data needs to be placed in packages, better approach than the current one will result in fewer packages.

One interesting feature of tasks is that they do not necessarily need to get the newest values of sensor data every time they execute. For each signal, it is determined how many times in a row, it can be lost or not sent in the network. This parameter is called *the number of allowed misses*, and we will refer to it in this way below. Since we are considering an ideal network, in which data cannot be lost, we will use this parameter to intentionally not package some signals. In this way, network occupancy can be significantly reduced.

FCIs represent nodes in this network-centric system. They calculate and create tables according to which they send data to the network. Therefore, the output of the algorithm should provide a

schedule for sending packets as well as their content.

5.6.2. Our approach

Depending on the need of the individual tasks, the signals within the FCI are collected in datasets. The FCI has information on the ideal reading period of these datasets. Based on this information, we can calculate a hyperperiod of reading (packing):

$$\text{hyperperiod} = LCM(T_1, \dots, T_n) \quad (14)$$

where T_i $i \in \{1, 2, \dots, n\}$ is ideal reading period of dataset i .

The hyperperiod represents the interval at which the entire schedule of packing and sending in the FCI is repeated. During this time, each dataset will be packed at least once and sent to the controller. It also covers all cases of sending different datasets at the same time. Therefore, the algorithm will calculate and use this interval in each FCI to determine its packages transmission schedule. Using the data in table 2, the hyperperiod of FCI1 is:

$$\text{hyperperiod} = LCM(50, 100, 150) = 300$$

Then we can determine time moments within hyperperiod in which each of data sets will be packed.

$$t_{S1-S20} = [50, 100, 150, 200, 250, 300]$$

$$t_{S15-S40} = [100, 200, 300]$$

$$t_{S41-S70} = [150, 300]$$

At any given time, the algorithm will attempt to pack the data so as to reduce network occupancy. The algorithm consists of two stages:

1. Determine all possible combinations of packing data for different controllers into the same package.
2. Perform bin packing algorithm to put as much data in as few packages as possible, taking into account the parameter of allowed misses.

The first part of the algorithm is illustrated in Figure 14.

Figure 14: Illustrative representation of the first phase of the algorithm

For each time in which the packing and sending of data in the network take place, the algorithm performs all possible combinations of packing. The number of combinations is determined by the number of different datasets being sent at that time point. Since we consider a medium-sized plant model and FCIs do not supply a large number of different controllers, it is possible to test all combination and thus find optimal solution. In large-sized plant model and bigger networks, this exhaustive search can lead to the algorithm running too long. In that case, one possible approach is to test a limited number of randomly selected combinations.

The second part of the algorithm performs packing. A modified BFD algorithm has been implemented for this purpose. Depending on the size, the signals ready to be packed at a certain point in time are arranged in descending order. Instead of packing all the signals, the algorithm first checks if it is necessary to send a signal and then puts it in a packet. Depending on the need for tasks, some signals may be omitted from the packet multiple times. After packing, the cost function is calculated:

$$f = \sum_{j=1}^n \sum_{i \in P(j)} L_j p_i \quad (15)$$

where n is the number of different paths, and L_j is the number of line segments that belong to the path $P(j)$ from FCI to the controller.

The type of package with the smallest value of the cost function is chosen as a best solution. This procedure is repeated for each FCI.

5.7. Architecture

The algorithm is implemented in the Python 3.8 with OOP paradigm. We used Python classes to model the system.

5.7.1. Network and devices

For network and device modeling, we created classes: Segment, Switch, Controller, FCI. Each of these elements has its ID represented as an integer and serves to distinguish elements in the network.

Segments are also specified with two nodes, which can be either switches or controllers. Each controller has a list of tasks assigned to it and the ID of the switch to which it is directly or indirectly connected via other controllers. Switches contain a list of all devices (FCIs and controllers) connected to them.

Each instance of the FCI class, in addition to the ID, also contains datasets it creates after reading the sensors. The list of packages and their contents, as well as the schedule for sending packages, are also filled in after the execution of the algorithm. FCIs later use this information for network routing purposes.

Figure 15 shows the network model.

Figure 15: Network model

5.7.2. Package, Signal, Task, Dataset

For the modeling of other important elements, we created the following classes: Package, Signal, Task, and Dataset.

Package instance is defined with its size, a list of signals it contains, its packing period, and a list of paths to intended controllers through a network. Appropriate methods for adding a signal into the package and assigning a packing period are created.

Classes Signal and Task have their IDs represented as integers, which serve to distinguish class instances. Signals are further specified with a number of allowed misses, and its size. Each instance of the Task class, in addition to the ID, contains its period, its execution time, and a dataset of signals the task processes.

Dataset instance contains the period, a list of signals in the dataset, and FCI and controller to which that dataset is allocated. There is also a list of time moments in which the dataset should be packed during the FCI hyperperiod and a path through which the dataset propagates in its packet from FCI to the controller that requires that dataset.

Figure 16 shows the interconnection between Package, Signal, Task and Dataset classes.

Figure 16: Package, Signal, Task and Dataset classes

6. Results

6.1. Implementation details

The algorithm is implemented using Python programming language. All network elements are modeled as classes. Input data are fed to the algorithm in the form of Excel datasheets. Three sheets correspond to 3 types of network devices (FCIs, Controllers, and Switches), and two sheets correspond to signals and tasks. Algorithm procedures read data from tables and initialize class instances with needed data. Once all needed class instances are created, the optimization procedures can run. The input data must be entered correctly in tables in the required format for the algorithm to work as expected.

The values of input data used for evaluation are:

- Allowed misses - random integer in range $[1, 5]$ for every signal
- Task period – period of the task in ms. Random integer between 100 and 500 with the step of 50.
- Task execution time – expressed in ms. Random number between 10 and 50 with the step of 10.
- Signals reading period – an ideal reading period of the set of signals allocated to the task expressed in ms. Defined as one-half of the task period signals are allocated to.

The specification of algorithm parameters used for evaluation is shown in table 3.

Parameter	Value
Execution time	9 s
Size of the packet's constant part	33 B

Table 3: Algorithm parameters

The execution time of 9 s is chosen because it is the biggest hyperperiod of all used FCIs and represents the smallest time unit in which the packing of all packets is repeated.

The output of the algorithm consists of:

- Time moments in which packages should be packed and sent for each FCI
- Contents of the packets which are packed and sent in computed time moments for each FCI
- Network occupancy of the segments
- Total network occupancy

Both the reference case and the proposed solution are performed, and results are exported to excel datasheets. Each excel file consists of $R + 1$ sheets. First R sheets represent time moments, sizes of the packets, and packets' contents that are sent from R FCIs in the network, where one sheet corresponds to the one FCI. In the last sheet are given network occupancy for every FCI and total network occupancy as a sum of occupancy of each network segment. An example of the packets in the excel datasheet is given in figures 17.

Time moments	Number of packets	Packet 1 size (B)	Packet 1 signals
75	1	585	[3, 4, 14, 2, 6, 7, 8, 9, 10, 11, 12, 1
100	1	582	[94, 96, 75, 77, 78, 81, 83, 84, 85,
150	2	585	[3, 4, 14, 2, 6, 7, 8, 9, 10, 11, 12, 1
175	1	843	[14, 21, 22, 28, 30, 10, 11, 12, 17,

Figure 17: Sending schedule of packages and their content

Algorithm is tested on three different topologies. All of them represent variants of the network of a medium-sized plant that consists of:

- 2400 input signals
- 80 control logic tasks
- 10 controllers
- 24 FCIs

Results for each topology are shown in the following subsections.

6.2. Topology 1

The first topology is depicted in the figure 18.

Figure 18: Topology 1

6.2.1. Reference case

Occupancy for each segment and the total network occupancy for reference case are given in the table 4.

Segments	Network occupancy (B)
Segment 1	991770
Segment 2	1264839
Segment 3	723399
Segment 4	3134502
Segment 5	4465863
Segment 6	320463
Segment 7	725607
Segment 8	2266524
Segment 9	1342896
Segment 10	2365158
Segment 11	859674
Segment 12	1054581
Segment 13	490680
Total network occupancy	20006136

Table 4: Network occupancy for the reference case (Topology 1, $H = 33B$, $T = 9s$)

6.2.2. Proposed solution

Occupancy for each segment and the total network occupancy for proposed solution are given in the table 5 when allowed misses are not taken into account.

Segments	Network occupancy (B)
Segment 1	988965
Segment 2	1264839
Segment 3	723399
Segment 4	3130377
Segment 5	4465401
Segment 6	320247
Segment 7	725145
Segment 8	2266524
Segment 9	1342896
Segment 10	2364696
Segment 11	857331
Segment 12	1054581
Segment 13	490680
Total network occupancy	19995081

Table 5: Network occupancy for the proposed solution without allowed misses (Topology 1, $H = 33B$, $T = 9s$)

Occupancy for each segment and the total network occupancy for proposed solution are given in the table 6 when allowed misses are taken into account.

Segments	Network occupancy (B)
Segment 1	315924
Segment 2	425202
Segment 3	211725
Segment 4	1003725
Segment 5	1421127
Segment 6	89190
Segment 7	220848
Segment 8	769824
Segment 9	495135
Segment 10	736698
Segment 11	279528
Segment 12	303513
Segment 13	185028
Total network occupancy	6457467

Table 6: Network occupancy for the proposed solution with allowed misses (Topology 1, $H = 33B$, $T = 9s$)

6.2.3. Comparison

It is noted that occupancy in the proposed solution is lower than occupancy in the reference case. Network occupancy reduction is given in the table 7 when packet header size is 33 B without allowed misses.

Segments	Absolute reduction (B)	Relative reduction (%)
Segment 1	2805	0.283
Segment 2	0	0
Segment 3	0	0
Segment 4	4125	0.132
Segment 5	462	0.010
Segment 6	396	0.124
Segment 7	462	0.064
Segment 8	0	0
Segment 9	0	0
Segment 10	462	0.020
Segment 11	2343	0.273
Segment 12	0	0
Segment 13	0	0
Total network	11055	0.055

Table 7: Network occupancy reduction without allowed misses (Topology 1, $H = 33B$, $T = 9s$)

Network occupancy reduction is given in the table 8 when packet header size is 33 B with allowed misses.

Segments	Absolute reduction (B)	Relative reduction (%)
Segment 1	675846	68.145
Segment 2	839637	66.383
Segment 3	511674	70.732
Segment 4	2130777	67.798
Segment 5	3044736	68.178
Segment 6	231453	72.184
Segment 7	504759	69.564
Segment 8	1496700	66.035
Segment 9	847761	63.129
Segment 10	1628460	68.852
Segment 11	580146	67.484
Segment 12	751068	71.220
Segment 13	305652	62.292
Total network	1354669	67.723

Table 8: Network occupancy reduction with allowed misses (Topology 1, $H = 33B$, $T = 9s$)

6.3. Topology 2

The second topology is depicted in the figure 19.

Figure 19: Topology 2

6.3.1. Reference case

Occupancy for each segment and the total network occupancy for reference case are given in the table 9.

Segments	Network occupancy (B)
Segment 1	991770
Segment 2	1264839
Segment 3	723399
Segment 4	2765916
Segment 5	4519197
Segment 6	320463
Segment 7	725607
Segment 8	2266524
Segment 9	1342896
Segment 10	1573671
Segment 11	859674
Segment 12	1054581
Segment 13	490680
Total network occupancy	18899397

Table 9: Network occupancy for the reference case (Topology 2, $H = 33B$, $T = 9s$)

6.3.2. Proposed solution

Occupancy for each segment and the total network occupancy for proposed solution are given in the table 10 when allowed misses are not taken into account.

Segments	Network occupancy (B)
Segment 1	988965
Segment 2	1264839
Segment 3	723399
Segment 4	2765916
Segment 5	4518339
Segment 6	320247
Segment 7	725145
Segment 8	2266524
Segment 9	1342896
Segment 10	1573209
Segment 11	857331
Segment 12	1054581
Segment 13	490680
Total network occupancy	18892071

Table 10: Network occupancy for the proposed solution without allowed misses (Topology 2, $H = 33B$, $T = 9s$)

Occupancy for each segment and the total network occupancy for proposed solution are given in the table 11 when allowed misses are taken into account.

Segments	Network occupancy (B)
Segment 1	315924
Segment 2	425793
Segment 3	212316
Segment 4	868707
Segment 5	1463913
Segment 6	90576
Segment 7	223860
Segment 8	769314
Segment 9	495219
Segment 10	517059
Segment 11	277848
Segment 12	303435
Segment 13	185028
Total network occupancy	6148992

Table 11: Network occupancy for the proposed solution with allowed misses (Topology 2, $H = 33B$, $T = 9s$)

6.3.3. Comparison

It is noted that occupancy in the proposed solution is lower than occupancy in the reference case. Network occupancy reduction is given in the table 12 when packet header size is 33 B without allowed misses.

Segments	Absolute reduction (B)	Relative reduction (%)
Segment 1	2805	0.283
Segment 2	0	0
Segment 3	0	0
Segment 4	0	0
Segment 5	858	0.019
Segment 6	396	0.124
Segment 7	462	0.064
Segment 8	0	0
Segment 9	0	0
Segment 10	462	0.029
Segment 11	2343	0.273
Segment 12	0	0
Segment 13	0	0
Total network	7326	0.039

Table 12: Network occupancy reduction without allowed misses (Topology 2, $H = 33B$, $T = 9s$)

Network occupancy reduction is given in the table 13 when packet header size is 33 B with allowed misses.

Segments	Absolute reduction (B)	Relative reduction (%)
Segment 1	675846	68.145
Segment 2	839046	66.336
Segment 3	511083	70.650
Segment 4	1897209	68.592
Segment 5	3055284	67.707
Segment 6	230067	71.752
Segment 7	501747	69.149
Segment 8	1497210	66.058
Segment 9	847677	63.123
Segment 10	1056612	67.143
Segment 11	581826	67.680
Segment 12	751146	71.227
Segment 13	305652	62.292
Total network	12750405	67.465

Table 13: Network occupancy reduction with allowed misses (Topology 2, $H = 33B$, $T = 9s$)

6.4. Topology 3

The third topology is depicted in the figure 20.

Figure 20: Topology 3

6.4.1. Reference case

Occupancy for each segment and the total network occupancy for reference case are given in the table 14.

Segments	Network occupancy (B)
Segment 1	991770
Segment 2	1264839
Segment 3	541440
Segment 4	2765916
Segment 5	3447690
Segment 6	320463
Segment 7	725607
Segment 8	2266524
Segment 9	923628
Segment 10	1320021
Segment 11	859674
Segment 12	1054581
Segment 13	490680
Total network occupancy	16973013

Table 14: Network occupancy for the reference case (Topology 3, $H = 33B$, $T = 9s$)

6.4.2. Proposed solution

Occupancy for each segment and the total network occupancy for proposed solution are given in the table 15 when allowed misses are not taken into account.

Segments	Network occupancy (B)
Segment 1	988965
Segment 2	1264839
Segment 3	541440
Segment 4	2765916
Segment 5	3446832
Segment 6	320247
Segment 7	725145
Segment 8	2266524
Segment 9	923628
Segment 10	1318140
Segment 11	857331
Segment 12	1054581
Segment 13	490680
Total network occupancy	16964268

Table 15: Network occupancy for the proposed solution without allowed misses (Topology 3, $H = 33B$, $T = 9s$)

Occupancy for each segment and the total network occupancy for proposed solution are given in the table 16 when allowed misses are taken into account.

Segments	Network occupancy (B)
Segment 1	315924
Segment 2	425793
Segment 3	213477
Segment 4	868707
Segment 5	1162998
Segment 6	90576
Segment 7	223860
Segment 8	769281
Segment 9	274125
Segment 10	442233
Segment 11	277848
Segment 12	303435
Segment 13	185028
Total network occupancy	5553285

Table 16: Network occupancy for the proposed solution with allowed misses (Topology 3, $H = 33B$, $T = 9s$)

6.4.3. Comparison

It is noted that occupancy in the proposed solution is lower than occupancy in the reference case. Network occupancy reduction is given in the table 17 when packet header size is 33 B without allowed misses.

Segments	Absolute reduction (B)	Relative reduction (%)
Segment 1	2805	0.283
Segment 2	0	0
Segment 3	0	0
Segment 4	0	0
Segment 5	858	0.025
Segment 6	396	0.124
Segment 7	462	0.064
Segment 8	0	0
Segment 9	0	0
Segment 10	1881	0.142
Segment 11	2343	0.273
Segment 12	0	0
Segment 13	0	0
Total network	8745	0.052

Table 17: Network occupancy reduction without allowed misses (Topology 3, $H = 33B$, $T = 9s$)

Network occupancy reduction is given in the table 18 when packet header size is 33 B with allowed misses.

Segments	Absolute reduction (B)	Relative reduction (%)
Segment 1	675846	68.145
Segment 2	839046	66.336
Segment 3	327963	60.572
Segment 4	1897209	68.592
Segment 5	2284692	66.267
Segment 6	230067	71.752
Segment 7	501747	69.149
Segment 8	1497243	66.059
Segment 9	649503	70.321
Segment 10	877788	66.498
Segment 11	581826	67.680
Segment 12	751146	71.227
Segment 13	305652	62.292
Total network	11419728	67.282

Table 18: Network occupancy reduction with allowed misses (Topology 3, $H = 33B$, $T = 9s$)

7. Discussion

7.1. Influence of different packing on network occupancy

In this subsection, we will try to answer the second research question:

RQ2: *How does the different packing of input data affect the network occupancy?*

Let us consider the reference case and the proposed solution without allowed misses from topology 1. In this topology, one dataset from FCI1 is assigned an ideal period of 75 ms, while another one is assigned an ideal period of 175 ms. FCI will pack and send signals only from these datasets at 525 ms, because in that time moment there are no other signals from other datasets scheduled for packing. The difference between the reference case and the proposed solution is that:

- In the reference case, two packets will be created at 525 ms
- In the proposed solution, one packet will be created at 525 ms

The algorithm tested both variants: with one packet and with two packets. In each variant, the algorithm calculated the network occupancy and decided in favor of the variant with less network occupancy. As variant with one packet produced less network occupancy than the variant with two packets, as shown in table 19, the algorithm decided to proceed with the creation of only one packet.

Case	Time moment (ms)	Number of packets	Packets' sizes (B)
Reference	525	2	843, 585
Proposed solution	525	1	1395

Table 19: Difference between reference and proposed solution in number of packets

The sum of packets' sizes in the reference case is 1428 B, which is higher than the packet size in the proposed solution for 33 B. Thus, in the proposed solution, network occupancy is reduced for the size of one header.

7.2. Influence of the BFD algorithm on package utilization

The BFD algorithm is chosen as a heuristic approach in the proposed solution. It achieves better results than the NF algorithm used in the reference case. A modified version of the BFD algorithm is implemented. Instead of packing all the signals, the algorithm first checks if it is necessary to send a signal and then places it in a packet.

Let us consider the reference case and the proposed solution without allowed misses from topology 1. In this topology, in the time moment, 1125 ms in FCI18 six packets are created in the proposed solution and six packets in the reference case. Difference in packet utilization is shown table 20.

Case	Time moment (ms)	Number of packets	Packets' sizes (B)
Reference	1125	6	1500, 1494, 552, 1332, 1431, 1221
Proposed solution	1125	6	1500, 1500, 546, 1332, 1500, 1152

Table 20: Difference between reference and proposed solution in the packets utilization

In the proposed solution, three packages are fully utilized, while one package is fully utilized in the reference case. Thus, there are more packages fully utilized in the proposed solution. Although the number of packages in this example is not reduced, the BFD algorithm has been shown to maximize package utilization. In larger networks, where a larger number of packages are sent at the same time, by applying BFD, we can also expect a reduction in the number of packages.

Signal	Allowed Misses	Size
S1	5	12
S2	3	15
S3	4	105
S4	3	105
S5	1	12
S6	3	15
S7	4	15
S8	3	15
S9	2	15
S10	5	15
S11	4	15
S12	4	15
S13	2	12
S14	3	105
S15	4	12
S16	5	12
S17	4	15
S18	2	12
S19	4	15
S20	1	15

Table 21: Allowed misses and size of signals in dataset S1-S20

Let us now consider how the parameter *Allowed Misses* affects the packaging. We will observe the packing of the S1-S20 dataset at three consecutive time points. The size and number of allowed misses are specified for each signal and shown in Table 21. Since the sending period of this dataset is 75ms, we are interested in time moments of 75ms, 150ms, and 225ms. Table 22 shows the content and timing of packets within FCI1 when data freshness is not considered. Table 23 shows the same results, but when the allowed misses are taken into account.

Time moment	Nr. of packets	Packet 1 size	Packet 1 signals
75	1	585	3,4,14,2,6,7,8,9,10,11,12,17,19,20,1,5,13,15,16,18
150	2	585	3,4,14,2,6,7,8,9,10,11,12,17,19,20,1,5,13,15,16,18
225	1	585	3,4,14,2,6,7,8,9,10,11,12,17,19,20,1,5,13,15,16,18

Table 22: Part of FCI1 packing schedule without allowed misses

Time moment	Nr. of packets	Packet 1 size	Packet 1 signals
75	1	585	3,4,14,2,6,7,8,9,10,11,12,17,19,20,1,5,13,15,16,18
150	1	405	71,72,73,68,69,74,70
225	1	72	20,5,13

Table 23: Part of FCI1 packing schedule with allowed misses

In the first case, each signal from the S1-S20 dataset is sent in every time moment. At 150ms, another dataset is sent along with this dataset, but its content is not displayed.

When the allowed misses are taken into account, the new values of each signal from the dataset are intentionally not sent at 150ms. Tasks executed in the controller process old signal values previously written in the registers. At 225ms, only three fresh signal values from the dataset (S5, S13, S20) are sent. The reason is that these signals are allowed not to be sent only once, i.e., tasks can only use the old value for one execution (see Table 21).

Even though we observed a very short time interval, we can conclude that this feature significantly reduces the network occupancy.

7.3. Influence of topology on network occupancy

In this subsection, we will try to answer the third research question:

RQ3: *How does the choice of the network topology affect the network occupancy?*

When formulating the problem and presenting a simple example, we noticed that the relative placement of elements in the network affects the occupancy of some segments in the network. Analyzing the obtained results, we can see that the segments that connect the switches are the most loaded. In our network model, these are segments 4, 5, and 10. This is expected because switches together with segments create a fundamental network tree. All other segments with controllers represent smaller branches of this tree. The idea is that just by moving certain elements to other switches tries to reduce the amount of data on these segments. It should be emphasized that there are also specific physical limitations among the elements in the network. The sensors are usually located in predetermined locations and cannot be deployed. Therefore, clusters of FCIs are formed that collect data from a specific group of sensors. FCIs belonging to one cluster cannot be moved to others. Therefore, moving any FCI to another location in the network in other topologies involves moving the entire cluster.

In topology 2, the occupancy of segments 4 and 10 is significantly reduced (see Table 24).

Segment	Topology 1	Topology 2
Segment 4	1003725 B	868707 B
Segment 5	1421127 B	1463913 B
Segment 10	736698 B	517059 B

Table 24: Difference between topology 1 and topology 2 in occupancy of segments 4, 5, and 10

This reduction is due to the shift of controllers 2 and 8 to switches 2 and 3, and the replacement of FCIs clusters on switches 1 and 3. In this way, the number of elements on external switches is reduced, thus reducing the number of devices sending or receiving data through segments 4 and 10. Although a significantly lower occupancy of segments 4 and 10 was achieved, the occupancy of segment 5 increased. This trade-off may be unacceptable because segment 5 is the busiest segment in the network.

Subsequent movements of the controller in topology 3 resulted in a reduced network occupancy of 14% compared to topology 1. With an additional reduction in the payload in segment 4, a significant difference is also noticeable in segment 5 (see Table 25).

Segment	Topology 2	Topology 3
Segment 4	868707 B	868707 B
Segment 5	1463913 B	1162998 B
Segment 10	517059 B	442233 B

Table 25: Difference between topology 2 and topology 3 in occupancy of segments 4,5, and 10

In the case of topology 3, we can see that the payload on segment 9 is also reduced. The occupancy of segment 8 in topologies 1 and 2 is equal to the sum of the size of data sent to controllers 4 and 5, while the occupancy of segment 9 is equal to the size of data sent only to controller 5. By a simple calculation, we get that a larger amount of data is sent to controller 5. Therefore, by replacing controllers 4 and 5, in topology 3, less occupation of segment 9 is achieved.

Adding new segments or switches can further reduce the payload in the network, but this would significantly change the network structure, and thus the results would not be comparable. The cost of new elements in the network should also be taken into account, and we leave that as future work.

8. Conclusions

In this paper, we addressed a multi-constrained network occupancy optimization problem, where a set of signals from different sensors have to be packed into a set of network packages. The signals are pre-grouped into datasets and, as such, assigned to the controllers. Tasks, running in the controllers, process these signals, and send the output to the actuators. Due to its specificity, this problem has not been deeply studied in the literature.

To solve this problem, we propose a heuristic approach which consists of two phases. The algorithm first determines all possible combinations of packing data for different controllers into the same package. The second phase represents the modification of the BFD algorithm, the offline heuristic algorithm used for solving bin-packing problems. In this phase, the packing procedure is performed. Based on the required data freshness in the tasks, the algorithm first verifies if it is necessary to send a signal, and then if it is, puts the signal in a package. Since this is an offline setting, it is possible to sort the signals by size before packing.

The reference case was also implemented, and results are obtained for both the reference case and the proposed solution to verify the effectiveness of the proposed algorithm. The reference case does not consider the freshness of the data and excludes any possibility of merging data for different controllers into the same packet. The packaging procedure is based on the NF algorithm. The use of the BFD algorithm in the proposed solution shows the increase in package utilization while merging datasets into the same packets has reduced network occupancy.

The algorithm was run on three topologies to see the influence of the relative placement of elements in the network. The obtained results showed that network occupancy could be significantly reduced by bringing the end nodes (controllers) closer to the source (FCIs). Sometimes it is not possible to do this because one controller can request data from remote sensors, i.e., from FCIs that do not belong to the same cluster.

As our approach currently does not make any considerations on the propagation time of the data, possible collisions in traffic, and part of the communication from controllers to actuators, future research may be conducted to optimize these issues.

8.1. Future work

Main directions for future work are:

- **Consider data transfer from controllers to FCIs (both way communication):**
Network-Centric control model models the propagation of data from FCIs to controllers and from controllers back to FCIs that is in both directions. This algorithm considers only one-way communication from FCIs to controllers. That is the reason why links between FCIs and switches are not considered. Optimizing these segments will only make sense if two-way communication is considered. Every FCI is directly connected to the switch, so data from FCIs are momentarily transferred to corresponding switches because the network is modeled as an ideal, and there are no data losses, while the propagation time is neglected. If two-way communication is considered, then links between FCIs and switches should be modeled as segments, and thus network occupancy optimized on them together with other segments.
- **Include "jitter" into calculations:**
As aforementioned, every signal has the ideal frequency by which it should be packed. With jitter taken into consideration, every ideal frequency would have the frequency range around it, which is defined by jitter. This means that the packing frequency of the signal can move in the jitter-defined frequency range. Further, this implies that the signal's ideal frequency can be changed to optimize the packing procedure and minimize network occupancy. The algorithm output, in addition to the aforementioned values, would contain new ideal frequencies for every signal.
- **Model a real network in which data can be lost, collisions may happen, and the propagation time of the data through the network is important:**
As the network is modeled as an ideal network, the loss of the data, eventual collisions, and propagation are neglected and not considered. This is not the case in the real world,

where each of these parameters impacts the network performance and eventually increases network occupancy. For future work, it would be feasible if a network model considers these parameters because the model would represent the real network better.

References

- [1] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu, “Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things,” in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, 2015, pp. 1–7.
- [2] “OPC Unified Architecture, Specification, Part 1: Overview and Concepts,” <https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-1-overview-and-concepts/>, 2017, [Online; accessed 14-May-2020].
- [3] I. Belai and P. Drahoš, “The industrial communication systems profibus and profinet,” *Applied Natural Sciences*, 01 2009.
- [4] W. Leinberger, G. Karypis, and V. Kumar, “Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints,” in *Proceedings of the 1999 International Conference on Parallel Processing*. IEEE, 1999, pp. 404–412.
- [5] O. Moneer, “Bin packing problem under multiple-criterions.”
- [6] G. Dósa and J. Sgall, “First fit bin packing: A tight analysis,” in *30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.
- [7] —, “Optimal analysis of best fit bin packing,” in *International Colloquium on Automata, Languages, and Programming*. Springer, 2014, pp. 429–441.
- [8] V. M. Lo, “Heuristic algorithms for task assignment in distributed systems,” *IEEE Transactions on computers*, vol. 37, no. 11, pp. 1384–1397, 1988.
- [9] K. Fleszar and K. S. Hindi, “New heuristics for one-dimensional bin-packing,” *Computers & operations research*, vol. 29, no. 7, pp. 821–839, 2002.
- [10] E. Hopper and B. C. Turton, “An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem,” *European Journal of Operational Research*, vol. 128, no. 1, pp. 34–57, 2001.
- [11] M. Salimi, A. Majd, M. Loni, T. Secoleanu, C. Secoleanu, M. Sirjani, M. Daneshtalab, and E. Troubitsyna, “Multi-objective optimization of real-time task scheduling problem for distributed environments,” in *Proceedings of the 6th Conference on the Engineering of Computer Based Systems*, 2019, pp. 1–9.
- [12] T. Grant, R. Janssen, and H. Monsuur, *Network Topology in Command and Control: Organization, Operation, and Evolution*. Information Science Reference, 2014.
- [13] “Industrial Communication,” https://www.pepperl-fuchs.com/global/en/classid_6416.htm, 2020, [Online; accessed 14-April-2020].
- [14] J. F. Kurose, *Computer networking: A top-down approach featuring the internet, 3/E*. Pearson Education India, 2005.
- [15] C. E. Spurgeon, *Ethernet: the definitive guide*. ” O’Reilly Media, Inc.”, 2000.
- [16] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC unified architecture*. Springer Science & Business Media, 2009.
- [17] “OPC Foundation,” <https://opcfoundation.org/>, 2020, [Online; accessed 7-May-2020].
- [18] R. D. Saraiva, N. Nepomuceno, and P. R. Pinheiro, “A layer-building algorithm for the three-dimensional multiple bin packing problem: a case study in an automotive company,” *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 490–495, 2015.

- [19] K.-H. Loh, B. Golden, and E. Wasil, "Solving the one-dimensional bin packing problem with a weight annealing heuristic," *Computers & Operations Research*, vol. 35, no. 7, pp. 2283–2291, 2008.
- [20] D. Pisinger and M. Sigurd, "The two-dimensional bin packing problem with variable bin sizes and costs," *Discrete Optimization*, vol. 2, no. 2, pp. 154–167, 2005.
- [21] L. Sheng, S. Xiuqin, C. Changjian, Z. Hongxia, S. Dayong, and W. Feiyue, "Heuristic algorithm for the container loading problem with multiple constraints," *Computers & Industrial Engineering*, vol. 108, pp. 149–164, 2017.
- [22] D. Liu, K. C. Tan, S. Huang, C. K. Goh, and W. K. Ho, "On solving multiobjective bin packing problems using evolutionary particle swarm optimization," *European Journal of Operational Research*, vol. 190, no. 2, pp. 357–382, 2008.
- [23] E. G. Coffman, Jr, M. R. Garey, and D. S. Johnson, "An application of bin-packing to multiprocessor scheduling," *SIAM Journal on Computing*, vol. 7, no. 1, pp. 1–17, 1978.
- [24] K. Bernhard and J. Vygen, "Combinatorial optimization: Theory and algorithms," *Springer, Third Edition, 2005.*, 2008.
- [25] G. Dósa, "The tight bound of first fit decreasing bin-packing algorithm is $FFD(I) \leq 11/9OPT(I) + 6/9$," in *International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*. Springer, 2007, pp. 1–11.
- [26] Z. Wang and K. Nip, "Bin packing under linear constraints," *Journal of Combinatorial Optimization*, vol. 34, no. 4, pp. 1198–1209, 2017.
- [27] J. F. Nunamaker Jr, M. Chen, and T. D. Purdin, "Systems development in information systems research," *Journal of management information systems*, vol. 7, no. 3, pp. 89–106, 1990.
- [28] "OPC Unified Architecture, Specification, Part 14: PubSub," <https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-14-pubsub>, 2018, [Online; accessed 13-May-2020].