Computer Science and Information Systems 20(1):251–276 https://doi.org/10.2298/CSIS211001008H

Multi-constrained Network Occupancy Optimization ⋆

Amar Halilovic[1], Nedim Zaimovic[2], Tiberiu Seceleanu[3], and Hamid Feyzmahdavian[4]

[1] Ulm University, Ulm, Germany
{amar.halilovic}@uni-ulm.de
[2] ALTEN AB, Västerås, Sweden
nedim.zaimovic@alten.se
[3] Mälardalen University, Västerås, Sweden
tiberiu.seceleanu@mdh.se
[4] ABB AB, Västerås, Sweden
hamid.feyzmahdavian@se.abb.com

Abstract. The greater the number of devices on a network, the higher load in the network, the more chance of a collision occurring, and the longer it takes to transmit a message. The size of load can be identified by measuring the network occupancy, hence it is desirable to minimize the latter. In this paper, we present an approach for network occupancy minimization by optimizing the packing process while satisfying multiple constraints. We formulate the minimization problem as a bin packing problem and we implement a modification of the Best-Fit Decreasing algorithm to find the optimal solution. The approach considers grouping signals that are sent to different destinations in the same package. The analysis is done on a medium-sized plant model, and different topologies are tested. The results show that the proposed solution lowers the network occupancy compared to a reference case.

Keywords: industrial control networking, packing algorithms, network performance.

1. Introduction

Increasingly different types of information are required in industrial processes and engineering systems, resulting in the diverse and widespread use of networks. Today, there are about 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 [31]. Thus, we can expect communications-capable machines to be the most common type of device in the future, leading to a considerable growth of the data volume. A significant increase of the network traffic and network load may limit and even decrease the network speed while increasing the data loss. Therefore, powerful data communication is necessary to handle this growth.

Network occupancy can show how large is the network load. Although network speeds are getting higher, it is always desirable to minimize the network occupancy, as data volumes and the number of related critical features are also increasing. Thus, network optimization plays a vital role in many applications that have certain speed and reliability requirements. Such requirements refer to accurate delivery of packages to their destination, in some guaranteed time frame.

⋆ Extension of the article published in the 7th Conference on the Engineering of Computer Based Systems (ECBS 2021), May 26–27, 2021, Novi Sad, Serbia.

Problem formulation. Our work is triggered by a move from the classical *Control-Centric System Model* towards a *Network-Centric Control Model* in the context of industrial automation systems (IAS). An IAS is concerned with the acquisition, delivery, and processing of input signals from sensors to controllers and delivery of the controller signals to actuators. Such a distributed system is usually deployed on the Open Platform Communications Unified Architecture (OPC UA) [1] protocol, running on top of Ethernet network technology. We discuss here about the part of the model between sensors and controllers. The reverse communication, plus additional - management related communication - can be covered by an extension of the proposed solution.

The sensors provide the process data to the control system. Their signals are aggregated in *Field Control Interface* devices (FCI) and grouped in larger, as possible, disjunctive datasets. FCIs in the network have registers for storing signal values. Each dataset has an *ideal* frequency at which the FCI assembles them into packets. However, for most of the signals, values packed at different than ideal time moments may still provide for good process management. On the other side of the network, there are controllers responsible for specific tasks. For each task, a dataset of signals is assigned to the process. These datasets overlap with datasets of signals sourcing at potentially different FCIs.

Further, *task allocation* is a problem that can be identified in multiple domains, and acknowledgment of the related complexity has long been on researchers' agendas. From a technical perspective, task allocation is considered an "essential problem" in the area of parallel computing [26], resurfacing with the advent of multi-core processors [15], and today also being "one of the most fundamental classes of problems in robotics" [4]. Moreover, scheduling task execution in distributed systems is, for a long time now known as an NP-hard problem [12]. Task allocation is also raising aspects of performance, the "optimality" of the action being one crucial step in the design of modern systems [28].

We identify our problem as both an allocation issue and a scheduling and performance issue. We study the above both considering an *ideal* network - where no packages are lost, as well as a "lossy" network, where a certain number of packages don't reach their intended destination. However, we do not propose here a solution for task allocation - there are several options to follow in the research literature, but, at the moment, we just want to illustrate the effect of this at the levels we discuss.

Our goal is to find a (close to) optimal packing schedule, maximize the packet utilization, thus minimizing the number of packets sent, so that the network occupancy, expressed in Bytes (B), is minimized.

Contribution. We consider to address the following topics:

- Determine a suitable optimization algorithm that minimizes network occupancy given multiple constraints.
- Investigate how different data packing methods affect the network occupancy in a network-centric model.
- Study the impact of network topology on the network occupancy in a network-centric model.
- Study the impact of different task-controller allocation on the network occupancy in a network-centric model.
- Study the impact of package loss on network occupancy and investigate how a suitable optimization algorithm handles package loss.

The considered constraints can be identified as: the size of the package, the assignments of datasets to tasks, the various timing specifications (reading times, packing times, expected arrival times), data updates.

Solution space. The first direction of study brings us to the rich body of research related to *bin packing* [16], [21], [8], [9]. 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 whose size varies.

Heuristic and meta-heuristic algorithms and approaches [18] 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. Alternative approaches are also studied for speeding-up bin packing solutions [10], [13].

Another direction is to investigate the use of various artificial intelligence approaches (*e.g.* genetic algorithms [24]), or more sophisticated approaches. These approaches can be applied to large-scale optimization problems, specifically if problem-specific algorithms do not yield satisfactory results. The main drawback of these algorithms is 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 sufficiently useful.

2. Background

Industrial plants are usually very complex systems, including a large number of devices such as sensors, controllers, and actuators. An important task is to connect all devices into a single infrastructure to make industrial communication work from the field level up to the management level [2]. The switching technology is one among many used in modern industrial networks, built with the support of devices such as hubs and switches. A hub forwards the packet to all its ports. When a packet enters a switch, the switch reads the Medium Access Control (MAC) address stored in the packet header, and, based on the value, it takes decisions on which of its ports to send the packet through.

Controller-Centric Systems. Devices in a *controller-centric architecture* - CCA (Fig. 1) - are directly connected to controllers, thus making controllers to "own" devices. Configuration data from the engineering to devices is deployed over controllers. Controllers not only focus on control logic execution, but also require certain knowledge about device-specific implementations. Routing of device information goes through the controllers.

Network-Centric Systems. The logical topology of the *network-centric architecture* - NCA (Fig. 1) - brings several benefits compared to CCA. As devices and controllers are logically on the same bus, any controller can use signals from any device. Thus, the system is more flexible. However, this also brings some drawbacks, one of which is the possible network congestion, especially in large systems.

The bin packing problem. Items of different volumes must be packed into a finite number of bins or containers each of a fixed given volume in a way that minimizes the number of bins used (Fig. 2).

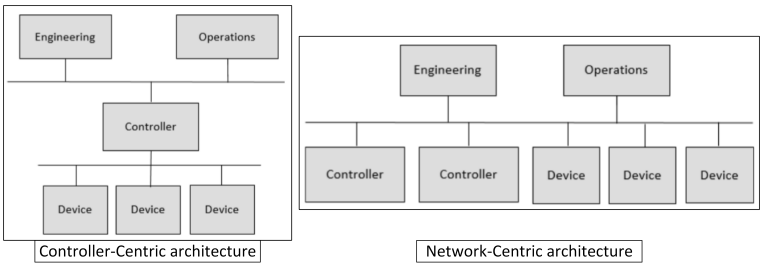


Fig. 1. Controller-Centric vs. Network-Centric Architectures

The total size of the items assigned to one bin should not exceed its capacity. This problem is known as **bin packing**, and it is a very popular challenge amongst the researchers across various domains. It comes in multiple variants, such as three-dimensional packing [25], packing by weight [19], and packing by cost [22].

The bin packing problem has many applications, such as filling up containers [27], loading trucks with weight capacity constraints [17], and job scheduling [6].

Heuristic algorithms. Bin packing is a NP-hard problem, and it is proven that no algorithm can achieve a performance ratio better than $\frac{3}{2}$ (unless $P = NP$) [5]. Heuristic methods are most commonly used to find the optimal solution to this problem.

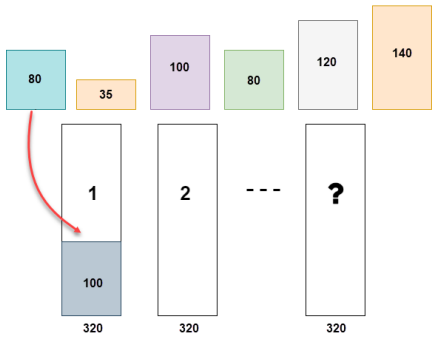


Fig. 2. The Bin packing problem.

One of the simplest heuristic algorithms for bin packing is probably *Next-Fit* (NF) [5]. 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 a new bin. A NF modification, *Next-k-Fit* (NkF) [20], holds k containers open instead of only one. *First-Fit* (FF) [8] is one of the most commonly used heuristic algorithms for solving the bin packing problem. Here, each item is placed into the first bin in which it will fit. If there is no such bin, it puts the item inside a new bin. Another well-known algorithm that provides a fast but often non-optimal solution is *Best-Fit* (BF) [9]. The idea is to put the item in the fullest bin in which it fits. *Worst-Fit* (WF) [5] 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. All these algorithms belong to a group of so-called *online* algorithms, and they consider items in an order defined by a list $\mathbf{L}$.

To measure the performance of online algorithms, 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 1 shows the upper bounds for each of the mentioned algorithms. Since the upper bounds on FF and BF algorithms are lower than those on NF and WF, the two most commonly used 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 [7]:

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

Note that the complexity of the Best-Fit Decreasing (BFD) algorithm is $\mathcal{O}(n \log n)$ for n objects, which is greater than the $\mathcal{O}(n)$ running time of the NF algorithm.

Table 1. Upper bounds of several algorithms.

Algorithm	**Upper bound**
Next-Fit	$NF(\mathbf{L}) \leq 2OPT(\mathbf{L}) - 1$ [5]
First-Fit	$FF(\mathbf{L}) \leq \lfloor 1.7OPT(\mathbf{L}) \rfloor$ [8]
Best-Fit	$BF(\mathbf{L}) \leq \lfloor 1.7OPT(\mathbf{L}) \rfloor$ [9]
Worst-Fit	$WF(\mathbf{L}) \leq 2OPT(\mathbf{L}) - 1$ [5]

3. Related Work

We divide this section on two of the main topics that we address in our study, *traffic optimization* and *package loss*. As observable in the next paragraphs, these are strongly correlated, but also independent solutions can be identified, in specific contexts.

Traffic optimization. Leinberger et al. [16] proposes 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 and amount of memory. The "bin" represents the parallel system, while the job wait queue is represented by an item list. The specific heuristics are though too heavy for the topic we address in this work.

In [27], the container loading problem with expiring orders is addressed. The authors classified this problem as a three-dimensional optimization problem with constraints such as orientation, stability, and loading priority. The items of an order must be entirely placed in the container or entirely be left behind. A heuristic algorithm handles first the expiring and then the non-expiring orders. The timing constraints make this approach not suitable in our context.

In [21], the bin packing problem is formulated as a multi-objective optimization problem: minimizing the bins used and minimizing the heterogeneousness of the elements in each bin. These two conflicting goals are formulated as a vector optimization problem. The authors emphasized the importance of trade-offs in this kind of optimization problem, which is the same we do here (by allowing some signals to not be sent over the network).

The bin packing problem under linear constraints is presented in [29], where the size of items to be packed is not given in advance. A modified Next-Fit algorithm is proposed. Linear programming computes the optimal value for size of items, and then the Next-Fit algorithm is deployed to solve the bin packing problem. The algorithm runs in polynomial time and has the same approximation ratio as CNF. The specifics of our application area - especially timing and package targeting - make the CNF not suitably.

A meta-heuristic approach for real-time task scheduling problems is employed in [24], guaranteeing end-to-end tasks' deadlines in distributed environments. Two different exploration scenarios are analyzed, single (looking for the minimal number of processing units for all the tasks) and multi-objective exploration (considering the total number of processing units and the end-to-end finishing time for all the jobs). Our problem is placed at a different level of granularity (tasks vs. packages), hence not benefiting enough from this approach.

Task allocation. There is a rich body of research focusing on the optimality of task allocation in networked environments. For on-chip networks, for instance, greedy network layer-based algorithms are found to be one solution [14]. Multiple criteria (distance to other nodes, energy levels, energy consumption, position) are analyzed when operating robotic networks [3]. A graph based framework is defined here for dynamically assigning tasks to set(s) of robots. For mesh networks, an interesting research [23] stresses the importance of architectures and algorithms, when considering the introduction of two new allocation algorithms.

We presented the above as just a (very) small part of what researchers focus on when discussing task allocation. We will not try to identify here a (new) allocation policy, but are illustrating the impact of allocation decisions on network occupancy.

Package loss. Several reasons are identified as leading to packages losses in modern day networked systems. As described in popular literature (e.g. [11]), some of the most important ones are *network congestion*, *defective hardware*, *outdated hardware and / or software*. Immediate solutions for the last two causes come from replacements and / or updates, respectively.

Packet loss is addressed at physical network levels [32], when some information is available on the forecasted data, or additional controls are necessary when such data is not available. Differently with the cited work, the transfers we intend to schedule are time-triggered and not event-triggered. Hence, we believe we do not need such a complex approach.

A new end-to-end congestion control algorithm is proposed in [30], based on a Naive Bayesian model. The approach analyzes both wired and wireless systems. The approach does apply priority features to the packets, and provides an identification of "where" (wired/wireless) the losses appear. The priority schemes are used to improve traffic conditions when the congestion is high. In our approach, we try to build the system such that congestion is controlled from the beginning, and thus packet losses are only very rare.

4. Building the approach

The *Network-Centric Control Model* (NCCM) allows us to model the acquisition and delivery of the input from sensors to controllers, their processing and the delivery of the controller signals to actuators on the *Smart Control Platform*. Fig. 3 depicts the NCCM, with the virtual paths that some packages travel.

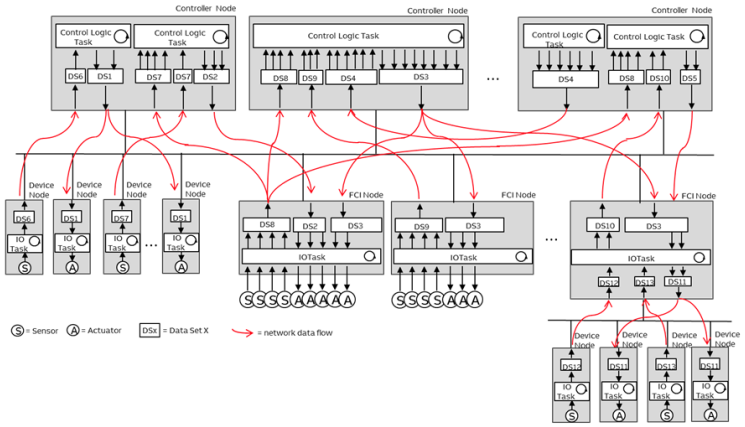


Fig. 3. Network-Centric Control Model

We focus here on the part of the model between sensors and controllers. Sensors are directly connected to the network bus, or via the FCIs - the approach we consider. Signals from assigned sensors are aggregated by the FCI in larger datasets to reduce the network load. Datasets in one FCI are disjunctive to datasets in any other FCI in the network. The time period in which the signal sends the data is negligible, and it is assumed that the FCI's registers contain the latest values of every assigned signal.

The network is modeled as an ideal system without delays. The network occupancy is defined as a number of bytes that flows through the network over a specified time interval. There is a number of switches that forward packets to the targeted controllers, and each switch is directly connected to at least one controller, and an FCI or another switch. Depending on the network topology, the number of network elements, their connections, and their disposition varies. At the other side of the network are controllers running tasks. Each task is defined by its period and execution time and is assigned a dataset of signals to process. These datasets overlap with datasets of signals in the FCIs.

The network package. The term *message* can have multiple meanings in networking. In this paper, we consider payload as a message. So, the data read from the sensor is a payload that is sent on 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 sent to the appropriate controller. The *DataSetMessage* is actually data from a single sensor, while the *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 Fig. 4-A.

The thus package consists, in addition to the payload, of different types of headers, security footer and signature. Everything except payload can be considered a fixed size. A simple representation of the resulting *abstract* package is shown in Fig. 4-B. 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.

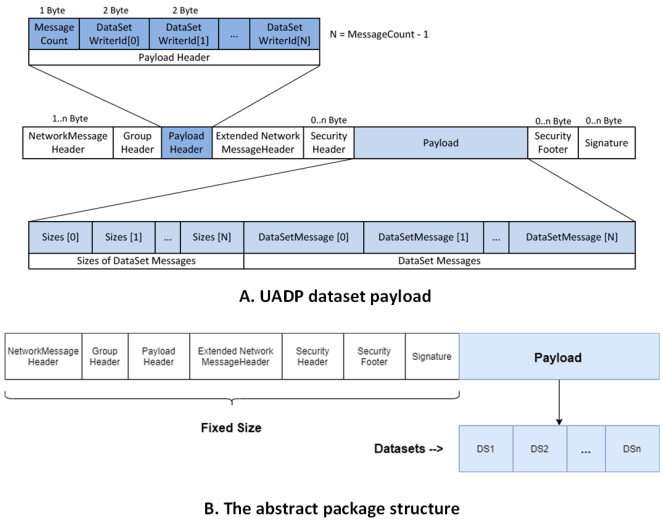


Fig. 4. A: The UADP dataset payload [1]; B: The abstract package structure.

Working example. In Fig. 5, we introduce a simple network topology, which we will use to present the algorithm. Each controller-controller, switch-switch, and the switch-controller connection is marked with a number. These numbers represent network lines / segments. Any information sent on the network travels some of these segments to its destination (one of the controllers). For example, let us 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. When a package is sent to controller 1, it only passes through segment 1. If any of the FCIs attached to switch 2 (SW 2) or switch 3 (SW 3) sends data to controller 1 or controller 2, this would occupy segments 1 and 3. It is expected that the segments connecting the three 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.

The mathematical model. The package size is defined by

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

where

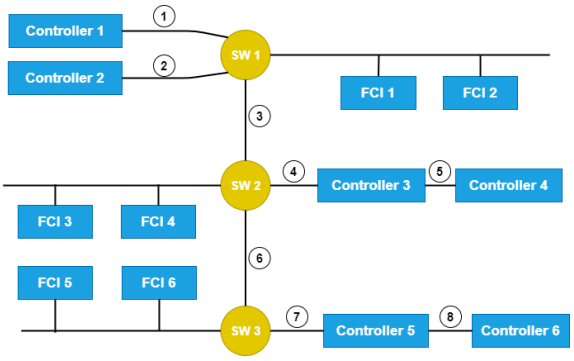


Fig. 5. Simple network

- $\sum_{i=1}^{n} x_{ik} = 1$ (each signal is only included in one package) with $x_{ik} = 1$ if signal k is put into package i and $x_{ik} = 0$ otherwise.
- p_i - the size of package i (bytes)
- H - the size of header and other constant (non-data) parts of the package
- DS_k - the data size of the signal k
- s - the number of signals inside the package
- n - the number of packages

Note that $p_i \leq 1500$ ensures that the total size of the loaded signals is not greater than the size of the package. Next, we try to optimize the network occupancy at a specific time interval (based on the repetitiveness of the signal activities). The relations below define segment occupancy as well as total network occupancy.

$$O_i = \sum_{j=1}^{n} \left\lceil \frac{T}{T_j} \right\rceil p_j, \; O_{tot} = \sum_{i=1}^{s} O_i$$

where

- O_i - Occupancy of the segment i
- O_{tot} - Total network occupancy
- s - The number of segments
- T - The time over which the network occupancy is observed
- 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
- p_j - Size of the package j
- n - The number of different packages that are sent through segment i

Packaging considerations. Consider next the same system of Fig. 5, where three controllers run each one task, with the characteristics illustrated in Table 2. The signals providing these datasets are attached to FCI1, and their ideal reading (and packing) periods are provided considering the Nyquist sampling theorem.

Table 2. Example of dataset assignment in FCI1

Task	Period (ms)	Controller	Dataset from sensors	Ideal reading period (ms)	FCI
T1	100	CTRL1	S1-S20	50	1
...	...	...	...	...	...
T3	100	CTRL3	S1-S20	50	1
T4	200	CTRL4	S15-S40	100	1
T5	300	CTRL5	S41-S70	150	1
...	...	...	...	...	...

Observe that, in the case of Controler 1 (CTRL1), all data does not fit into one package (as often is the case). Therefore, multiple packets are created, with the risk that some are not used to the maximum capacity. The packing mechanism advances 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 applies the NF algorithm - and we use it as a *reference case*. Fig. 6 shows the result of the NF algorithm applied to the data sent from FCI1 to CTRL1.

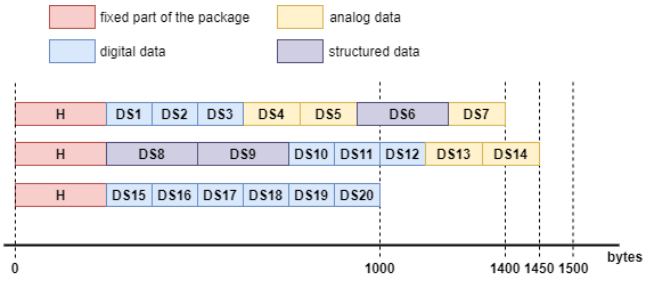


Fig. 6. Reference case packing procedure

Notice next the overlap in the data sampling for sensors S15 to S20, required basically every 100ms, by T3 and every 200ms, by T4 - hence read and ideally packed every 50ms.

If the respective data is packed in different packages (for T3 and T4, respectively), we can see that these datasets will travel through segments 3 and 4 at the same time, even though they contain the same information. This will result in an increase of the occupancy of these segments. However, data may be sent together in the same package to reduce network occupancy, with some timing, and / or package size penalties (in general, when periods don't match into some mathematical integer relation).

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-offs 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 network occupancy.

The proposed approach. As seen above, often datasets from the same FCI are sent to different controllers at the same time. As a result, some network lines may become congested. Given that there are dozens of such FCIs in the network, the load on the respective 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, with some potential minor drawbacks - also the occupation of leaf branches of the network would increase. This is a trade-off that must be taken into account when solving this issue.

The packing algorithm in the reference case leaves the packages unfilled. This can lead to unnecessary splitting of the dataset into multiple packages.

One interesting feature of task execution is that it does not require to run on the newest values of sensor data every time. For each signal (and corresponding task), it is specified how many times in a row data can be lost (or not sent in the network). We call this parameter the number of *allowed misses*, referred as such henceforth. 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.

The algorithm. The goal of the algorithm is to minimize the number of packets in the network. We describe here the implementation of the algorithm.

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):

$$hyperperiod = LCM(T1, \ldots, Tn),$$

where $T1, \ldots, Tn$ are the ideal reading periods of respective datasets.

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 the respective package-transmission schedule. Using the data in table 2, the hyperperiod of FCI1 is:

$$hyperperiod = LCM(50, 100, 150) = 300$$

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

$$\begin{aligned} t_{S1-S20} &= [50, 100, 150, 200, 250, 300] \\ t_{S15-S40} &= [100, 200, 300] \\ t_{S41-S70} &= [150, 300] \end{aligned}$$

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 Fig. 7. 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 combinations and thus find optimal solution. In large-sized plant models and wider networks, this exhaustive search can lead to the algorithm running for too long. In that case, one possible approach is to test a limited number of randomly selected combinations.

The second part of the algorithm addresses 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 requirements from tasks, some signals may be omitted from the packet for several consecutive times.

After packing, the we compute the cost function:

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

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 cost function basically calculates the network occupancy caused by a particular packet type. The type of package with the lowest value of the cost function is chosen as a best solution. This procedure is repeated for each FCI.

Package loss handling. The presented algorithm calculates the packing schedule for each FCI before commissioning. The resulting schedule implies that each packet will be sent at a specific time of sending and thus allow the tasks to operate with new data when needed. Losing a packet can result in tasks operating on old data or even task failure. In time and safety-critical systems, this temporary disruption of the proper operation is a potentially serious hazard. In addition, the network occupancy increases due to the data to be re-transmitted to the controllers. The classic approach to addressing this issue is to resend the lost packets in the next sending period. Our strategy for package loss handling is based on the packaging algorithm itself. Once a lost packet is identified, the FCI will associate it with packages that are sent at the next sending time according to a schedule. All datasets contained in the packages are re-grouped into new packages in the manner described in the algorithm. The FCI packing schedule is then updated for the next sending time. There are many advantages to this approach over classic re-sending.

Consider the case where the same packet is sent in two consecutive time moments as shown in Fig. 8. The packet P1 that is lost at time kT is marked in red, where T is the packet sending period. At the moment $(k+1)T$, the result of both approaches is shown, where the classical approach is marked in yellow and our strategy in blue.

On can notice that re-sending the packet will result in two of the same packets being sent to the same location (the re-sent packet is highlighted in green). The proposed algorithm identifies the same data in two packets and decides not to send them again. In this way, network congestion can be reduced.

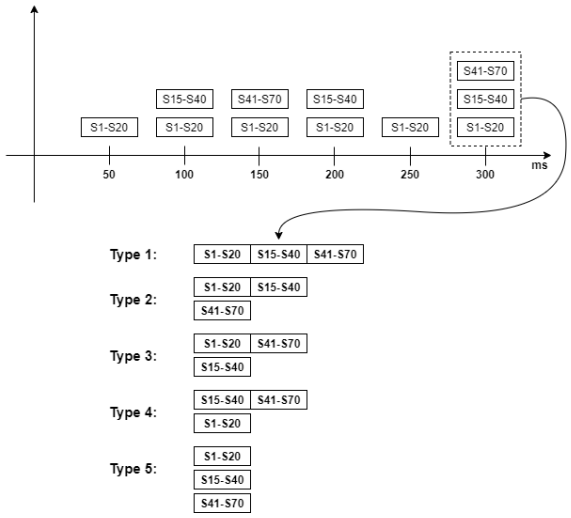


Fig. 7. Representation of the first phase of the algorithm

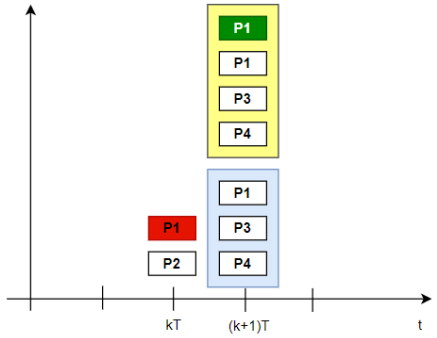


Fig. 8. Re-sending the same package.

Some systems are designed in such a way that devices that collect and package data send new data at each time of packaging. This avoids re-sending the packages and prevents the issue presented above. The disadvantage of such a design is that data is sent to controllers/tasks even when they do not need fresh data.

Apart of not sending the same data, the algorithm tries to merge datasets into the same packets. This is illustrated in Fig. 9, where Fig. 9.a shows part of the schedule of one FCI for two consecutive time moments, while Fig. 9.b and 9.c compare different approaches after losing a package with dataset DS1.

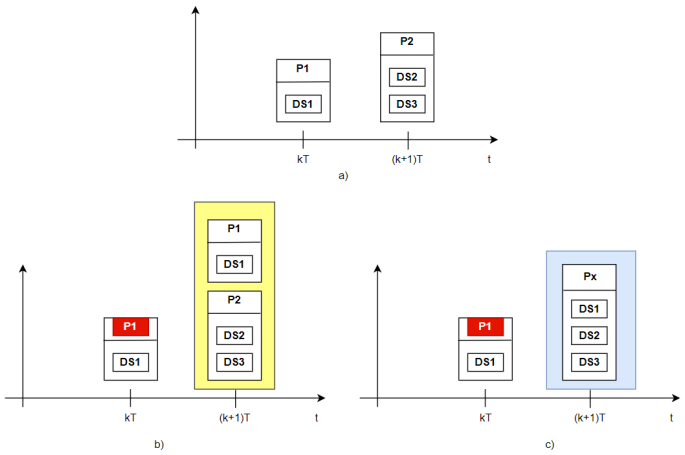


Fig. 9. a) No package loss case. b) "Re-send" approach after package loss. c) Proposed solution with merging datasets into a new package.

If calculated as a better option the algorithm will create a new package and not just re-send the lost one. The newly created package contains the data of the lost package and the packages scheduled at the time of sending. The schedule is being updated and the goal is to reduce network occupancy.

There is one drawback to this approach. If the new packet is lost, it will result in the loss of more data than in the case of just re-sending the packet. Also more controllers will be affected and left without fresh data. However, taking into account that the expected packet loss in the network is less than 10% the observed impact of this drawback is negligible.

In real-time systems, it is very important that tasks are executed using fresh data. Missing a deadline can lead to task failure and, in some cases, catastrophic consequences. Consider the case of 2 packages, P1 and P2 sent with different periods $T1$ and $T2$. Suppose a task that receives package P1 executes also with period $T1$. For the task to be executed in real-time it is necessary that a new data contained in package P1 is delivered before task starts to execute. In an ideal network that would always be fulfilled. However, in case package P1 gets lost it must be sent again before time period $T1$ expires. Using

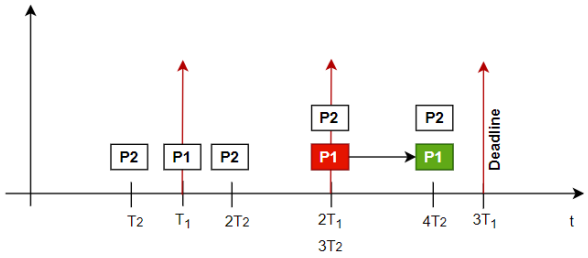


Fig. 10. Package delivery in time

our approach, packet P1 will be sent the next time any packet is sent from the FCI. In the case shown in Fig. 10 packet P1 is sent together with packet P2 before the deadline.

In general, it is guaranteed that package Pi with period Ti will arrive on time in case of loss if there is a package or set of packages that have a sending period less than Ti.

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

For network and device modeling, we created classes: Segment, Switch, Controller, FCI. Each of these elements has an ID represented as an integer and which 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.

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

A *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. Methods are defined for adding a signal into the package and assigning a packing period.

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.

A *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.

The above contribute, together with algorithm specific procedures, to the realisation of a small tool, with the interface shown in Fig. 11. At this moment the files used for input are hard-coded. The application expects the introduction of the execution time, and can run both the reference case (NF algorithm for the ideal network case without data

losses) and also the proposed solution for both the ideal network case and the case with data losses, calculating the network occupancy and allowing saves into Microsoft Excel.

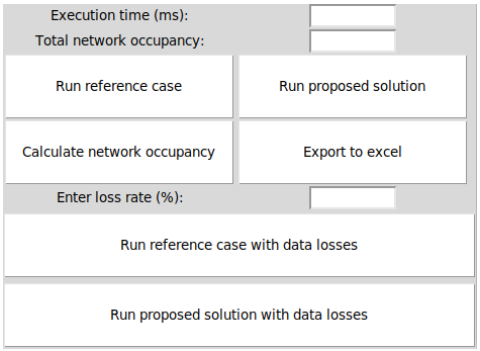


Fig. 11. The application GUI.

5. Simulation Results

To validate our working example, we apply the algorithm on a medium size plant use case, containing 2400 sensors (and associated signals), 80 control tasks, 24 FCIs and 10 controllers.

The input data is 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. The program reads data from tables and initialize class instances with needed data. Once all needed class instances are created, the optimization procedures can run. We organize the following based on 2 perspectives of the network operation.

1. *ideal*: No packages are lost. This is meant to help us create a picture of what benefits we can extract from our approach without a more complex context;
2. *real*: A "normal" loss of packages is considered and implemented by the simulation tool.

I both the above situation, the input data must be entered correctly in tables in the required format for the algorithm to work as expected. The 2^{nd} perspective above requiring registration of the additional information concerning the percentage of data loss across the network, which we modeled as an input to our graphical tool.

As shown in Fig. 11, a user can input the desired loss percentage as an integer. Below that are the buttons for running reference and our approach considering package losses.

Ideal network operation. The algorithm is tested on three different topologies (T_1. T_2, T_3), illustrated in Fig. 13. In all topologies the network is composed of 13 segments. 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.

We chose an execution time of 9s, because it is the largest hyperperiod of all used FCIs and represents the shortest time unit after which the packing of all packets is repeated. The output of the algorithm consists of:

- The time moments in which packages should be packed and sent for each FCI
- The content of the packets which are packed and sent in computed time moments for each FCI
- The network segment occupancy
- The total network occupancy

Both the reference case and the proposed solution are executed, and results are exported to Excel datasheets, each file consisting of $R + 1$ sheets. First R sheets represent time moments, sizes of the packets, and packets' content that are sent from the R FCIs in the network (one sheet per FCI). The last sheet contains the 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 datasheet is given in Fig. 12.

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,

Fig. 12. Sending schedule of packages and their content

Results for each topology are shown in Table 3, considering the packet header size 33B. The occupancy in the proposed solution is lower than the occupancy in the respective reference. The allowance and the selection of "no-send packages" is application and system specific, and is defined by system owners or designers - see also Table 5, further down.

Considering package losses. The algorithm is tested on topology 1 (Fig. 13) of the medium-sized plant without the allowance and selection of "no-send packages", described by the parameter "Allowed misses", as we focus our attention here on strategies to deal with package losses leaving aside the impact and discussion of different network topologies and allowance of "no-send packages" for the case of the ideal network.

As shown in Fig. 11, a user can enter loss percentage, after which one of the algorithms ("reference case" or "proposed approach" - described in detail in the previous section) is called. The lost packets are chosen randomly from the sending schedule created by the proposed solution without data losses until the limit of lost packages is reached, defined by

Table 3. Network occupancy relative reduction, on each topology (T_1, T_2, T_3).

	No misses allowed (%)			**Misses allowed** (%)		
Segments	T_1	T_2	T_3	T_1	T_2	T_3
Segment 1	0.283	0.283	0.283	68.145	68.145	68.145
Segment 2	0	0	0	66.383	66.336	66.383
Segment 3	0	0	0	70.732	70.650	60.572
Segment 4	0.132	0	0	67.798	68.592	68.592
Segment 5	0.010	0.019	0.025	68.178	67.707	66.267
Segment 6	0.124	0.124	0.124	72.184	71.752	71.752
Segment 7	0.064	0.064	0.064	69.564	69.149	69.149
Segment 8	0	0	0	66.035	66.058	66.059
Segment 9	0	0	0	63.129	63.123	70.321
Segment 10	0.020	0.029	0.142	68.852	67.143	66.498
Segment 11	0.273	0.273	0.273	67.484	67.680	67.680
Segment 12	0	0	0	71.220	71.227	71.227
Segment 13	0	0	0	62.292	62.292	62.292
Total network	0.055	0.039	0.052	67.723	67.465	67.282

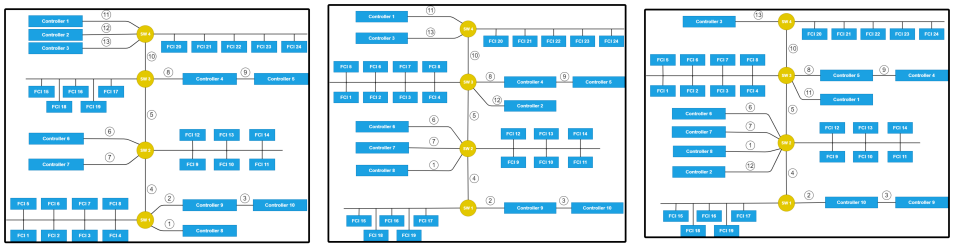


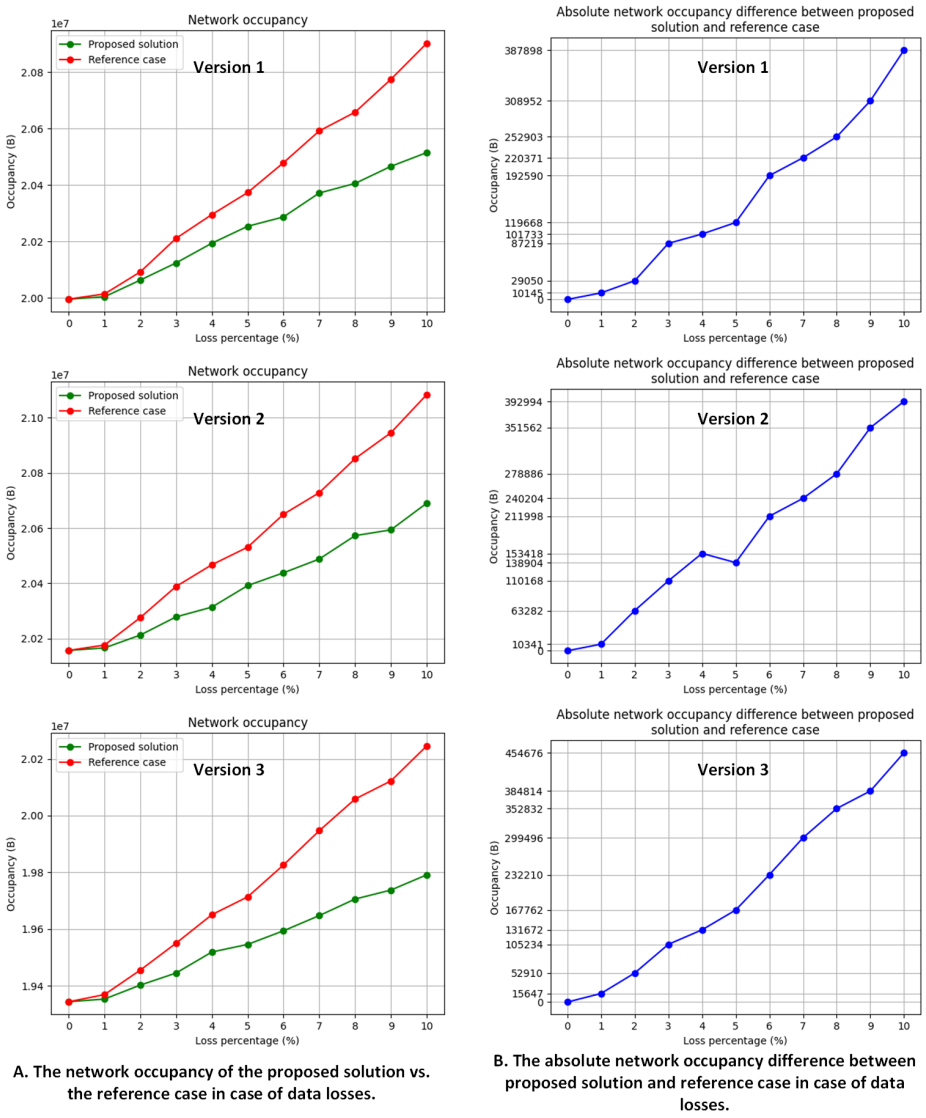
Fig. 13. Topologies

the entered loss percentage. Consequently, the calculation occupancy of network segments and total network is computed, and results are exported together with the packing schedule for every FCI as an Excel file.

An acceptable (depending on the application) loss rate in networks should be below 10%. With respect to that, we run our experiments with integer loss percentage rates from 1% to 10% including both limits. For every loss percentage rate from the closed interval $[1\%, 10\%]$ we run reference case and proposed solution 30 times each so that we have enough samples for treating results as a Gaussian distribution model. For every particular sample, we calculate occupancies of segments and the total network. Here we present averaged results. We test results on three different versions of the topology 1 (Fig. 13). The first version is one also used in the experiments without data losses, while the other two are created by different allocations of tasks to controllers, as discussed further down.

We plot in Fig. 14 the network occupancy change with respect to loss percentage for both the reference case and all the discussed versions. The right side column in the same Fig. shows the absolute network occupancy difference change with respect to loss per-

centage between the proposed solution and the reference case for three different versions of topology 1 (Fig. 13).



A. The network occupancy of the proposed solution vs. the reference case in case of data losses.

B. The absolute network occupancy difference between proposed solution and reference case in case of data losses.

Fig. 14. Simulation results considering packet losses. Column A: Network occupancy of the proposed versions and the reference case. Column B: Absolute network occupancy difference between the proposed versions and the reference case.

Impact of task allocation. In the following, we observe the task allocation impact on the network occupancy, in the context of topology 1. We consider the impact on the already

optimized solution, which we use as a reference. Differently from the already analyzed system, we allocate now the tasks such that on any given controller, there are either tasks which process signals with at least two different reading periods (if these are among the low values), or the tasks process relatively many signals with high reading periods, as the system *version 2*. The *version 3* of the system has the tasks allocated based on their respective signal reading period.

We run the simulation of the system versions, both in the ideal situation as well as considering losses, and the results are presented in Fig. 15.

Segments	Network occupancy per segments (B)		
	Version 1	Version 2	Version 3
Segment 1	988965	537603	891954
Segment 2	1264839	2343096	371304
Segment 3	723399	1689150	218160
Segment 4	3130377	3009135	2527425
Segment 5	4465401	3761547	4193820
Segment 6	320247	421734	555987
Segment 7	725145	539094	644262
Segment 8	2266524	613284	1453329
Segment 9	1342896	197775	776817
Segment 10	2364696	3528786	3656628
Segment 11	857331	2325630	916380
Segment 12	1054581	981789	1750530
Segment 13	490680	208656	1386711
Total network occupancy	19995081	20157279	19343307

Fig. 15. Network occupancy in the 3 analyzed system variants, no package losses.

6. Discussion

For studying how the different packing of input data affect the network occupancy, let us consider the reference case and the proposed solution without allowed misses, running on topology 1. Here, one dataset from FCI1 is assigned an ideal period of 75ms, while another one is assigned an ideal period of 175ms. FCI will pack and send signals only from these datasets at 525ms, 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:

- Reference case: two packets will be created at 525ms
- The proposed solution: one packet will be created at 525ms.

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 the variant with one packet produced less network occupancy than the variant with two packets, as shown in table 4, the algorithm decided to proceed with the creation of only one packet.

The sum of packet sizes in the reference case is 1428 B, which is higher than the packet size in the proposed solution by 33 B, corresponding to one less header size.

Table 4. Reference vs. proposed solution.

Case	Time moment (ms)	Number of packets	Packets' size (B)
Nr. of packets			
Reference	525	2	843, 585
Proposed solution	525	1	1395
Packet utilization			
Reference	1125	6	1500, 1494, 552, 1332, 1431, 1221
Proposed solution	1125	6	1500, 1500, 546, 1332, 1500, 1152

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 of the described use case, without allowed misses, running on topology 1. At time 1125ms, at FCI18, we have the packet utilization as shown in Table 4. In the proposed solution, three packages are fully utilized, while only one package is fully utilized in the reference case. 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. This also shows us that using "plain" bin packing algorithms is not sufficient. They require modification and combination with other algorithms to achieve a significant reduction in packets in the network.

Impact of the *Allowed Misses* parameter. We observe here 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 5. Since the sending period of this dataset is 75ms, we are interested in time moments of 75ms, 150ms, and 225ms. Table 6 shows both the content and timing of packets within FCI1 when data freshness at controllers is not considered (always required to have new data), and when the allowed misses are taken into account.

Table 5. Allowed misses and signal (S1-S20) data-size

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

When the allowed misses are taken into account, at 150ms, the new values of some signals from the dataset are not sent, as planned by the algorithm. At such moments, the affected tasks will process the signal values previously received and saved in specific registers of the controllers. At 225ms, only three fresh signal values from the dataset (S5, S13, S20) are sent. The reason is that these signals are allowed to skip transmission only

once, as the receiving tasks can use an old value of the signals for one cycle only (see Table 5).

Table 6. Part of FCI1 packing schedule.

Time moment	Nr. of packets	Packet 1 size	Packet 1 signals
No misses allowed			
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
Misses allowed			
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

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

Influence of topology on network occupancy. 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 7). 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 segments 5 increased. This trade-off may be unacceptable because segment 5 is the busiest segment in the network.

Table 7. Occupancy of segments 4, 5, and 10: T_1 vs. T_2

Segment	T_1	T_2
Segment 4	1003725 B	868707 B
Segment 5	1421127 B	1463913 B
Segment 10	736698 B	517059 B

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 8).

Table 8. Difference between T_2 and T_3 in occupancy of segments 4,5, and 10

Segment	T_2	T_3
Segment 4	868707 B	868707 B
Segment 5	1463913 B	1162998 B
Segment 10	517059 B	442233 B

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.

Considering package losses. Analyzing the plots in Fig. 14 A and B, it results that the proposed solution achieves lower network occupancy than the reference case with data losses. Additionally, it is visible that the absolute difference between the proposed solution and the reference case increases with the loss percentage - as expected. Table 9 presents percentage-wise the relative differences between the proposed solution and the reference case. The relative difference increases monotonically with the loss percentage increase. Even if these relative differences are not high percentage-wise, given the huge amount of data in real industrial networks, there will still be achieved a high absolute reduction in network occupancy.

Impact of task allocation. We refer in the following to the numbers presented in Fig. 15 and in Table 9. We notice, thus, that *version 3* of the task allocation offers a 3+% decrease of the overall traffic, while *version* 2 increases the traffic by a 0.8%. At the same time, traffic on some segments vary in much larger amounts.

It is important to remind here that the allocation versions are selected on a semi-random basis. The fact stands to only show that a good algorithm for allocation of tasks, would be a good complement to the work described in this report.

Table 9. Network occupation: *Version 1* (reference) vs. *Version 2* and *Version 3*.

	Differences in network occupation (%)													
Segment	1	2	3	4	5	6	7	8	9	10	11	12	13	Total
Version 2	-45,6	85,2	133,5	-3,9	15,8	31,7	-25,7	-72,9	-85,3	49,2	171,3	-6,9	-57,5	0,8
Version 3	9.8	-70,6	-69,8	-19,3	-6,1	73,6	-11,2	-35,9	-42,2	54,6	6,9	66,0	182,6	-3,3

7. Conclusions

We addressed here 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.

We proposed a heuristic approach over two phases. The algorithm first determines all possible combinations of packing data for different destinations into the same package. The second phase is based on a modification of the BFD algorithm. Based on the tasks' requirements on data freshness, the algorithm verifies the necessity of sending a signal, and only if so, it assigns the signal to a package. A reference case was also implemented, according to current industrial practices. 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 same packets has reduced network occupancy. The obtained results showed that network occupancy could be significantly reduced by bringing the end nodes closer to the sources. This is not always possible, as controllers may require data from variously placed FCIs. We showed that our approach achieves lower network occupancy in case of data losses and that different task-controller allocations can additionally increase or decrease network occupancy. A good algorithm for allocation of tasks would a good addition to our algorithm.

Future work. Our approach currently did not include several aspects of networked communication, such as propagation time and communication from controllers to actuators. Future research will be conducted to include these aspects. Note also that we only considered one-way communication, but a bi-directional perspective is necessary. However, the same approach can be extended to cover controllers as sources and FCIs as destinations - though with a reduced set of options for packing, as the number of signals at controllers (as sources) is considerably lower than what we see in the other direction. When two-way communication is considered, links between FCIs and switches should be modeled as segments.

References

1. OPC Unified Architecture, Specification, Part 14: PubSub. `https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-14-pubsub` (2018), [Online; accessed 13-May-2020]
2. Industrial Communication (2020), `https://www.pepperl-fuchs.com/global/en/classid_6416.htm`
3. Alirezazadeh, S., Alexandre, L.A.: Dynamic task allocation for robotic network cloud systems. The Intl. Conf. on Parallel and Distributed Processing with Applications, Big Data and Cloud Computing, Sustainable Computing and Communications, Social Computing and Networking pp. 1221–1228 (2020)
4. Aziz, H., Chan, H., Cseh, Á., Li, B., Ramezani, F., Wang, C.: Multi-robot task allocation – complexity and approximation. In: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems. pp. 133—-141 (2021)
5. Bernhard, K., Vygen, J.: Combinatorial optimization: Theory and algorithms. Springer, Third Edition, 2005. (2008)
6. Coffman, Jr, E.G., Garey, M.R., Johnson, D.S.: An application of bin-packing to multiprocessor scheduling. SIAM Journal on Computing 7(1), 1–17 (1978)
7. Dósa, G.: 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. pp. 1–11. Springer (2007)
8. Dósa, G., Sgall, J.: 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)

9. Dósa, G., Sgall, J.: Optimal analysis of best fit bin packing. In: International Colloquium on Automata, Languages, and Programming. pp. 429–441. Springer (2014)
10. Fleszar, K., Hindi, K.S.: New heuristics for one-dimensional bin-packing. Computers & operations research 29(7), 821–839 (2002)
11. FMS, P.: Packet loss: problems, causes and solutions in 2020 (2020), https://pandorafms.com/blog/packet-loss.
12. GAREY, M.R., JOHNSON, D.S.: Complexity results for multiprocessor scheduling under resource constraints. In: Proceedings of the 8th Annual Princeton Conference on Information Science and Systems (1974)
13. Hopper, E., Turton, B.C.: An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem. European Journal of Operational Research 128(1), 34–57 (2001)
14. Jiang, Y., Zhou, Y., Li, Y.: Network layer-oriented task allocation for multiagent systems in undependable multiplex networks. In: The 25th International Conference on Tools with Artificial Intelligence. pp. 640–647. IEEE (2013)
15. Kim, S.I., Kim, J.K., Ha, H.U., Kim, T.H., Choi, K.H.: Efficient task scheduling for hard real-time tasks in asymmetric multicore processors. Lecture Notes in Computer Science 7440, 187–196 (2012)
16. Leinberger, W., Karypis, G., Kumar, V.: Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints. In: Proceedings of the 1999 International Conference on Parallel Processing. pp. 404–412. IEEE (1999)
17. Liu, D., Tan, K.C., Huang, S., Goh, C.K., Ho, W.K.: On solving multiobjective bin packing problems using evolutionary particle swarm optimization. European Journal of Operational Research 190(2), 357–382 (2008)
18. Lo, V.M.: Heuristic algorithms for task assignment in distributed systems. IEEE Transactions on computers 37(11), 1384–1397 (1988)
19. Loh, K.H., Golden, B., Wasil, E.: Solving the one-dimensional bin packing problem with a weight annealing heuristic. Computers & Operations Research 35(7), 2283–2291 (2008)
20. Mao, W.: Tight worst-case performance bounds for next-k-fit bin packing. SIAM Journal on Computing 22(1), 46–56 (1993)
21. Moneer, O.: Bin packing problem under multiple-criterions, https://www.cse.huji.ac.il/~ai/projects/old/binPacking2.pdf, [Accessed: 9-July-2020]
22. Pisinger, D., Sigurd, M.: The two-dimensional bin packing problem with variable bin sizes and costs. Discrete Optimization 2(2), 154–167 (2005)
23. Postawka, A., Koszałka, I.: Task allocation within mesh networks: Influence of architecture and algorithms. In: Selvaraj, H., Zydek, D., Chmaj, G. (eds.) Advances in Intelligent Systems and Computing. vol. 366, pp. 869–875. Springer, Cham (2015)
24. Salimi, M., Majd, A., Loni, M., Seceleanu, T., Seceleanu, C., Sirjani, M., Daneshtalab, M., Troubitsyna, E.: 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. pp. 1–9 (2019)
25. Saraiva, R.D., Nepomuceno, N., Pinheiro, P.R.: A layer-building algorithm for the three-dimensional multiple bin packing problem: a case study in an automotive company. IFAC-PapersOnLine 48(3), 490–495 (2015)
26. Schoneveld, A., De Ronde, J., Sloot, P.: On the complexity of task allocation. Complexity 3(2), 52–60 (1997)
27. Sheng, L., Xiuqin, S., Changjian, C., Hongxia, Z., Dayong, S., Feiyue, W.: Heuristic algorithm for the container loading problem with multiple constraints. Computers & Industrial Engineering 108, 149–164 (2017)
28. Wang, S., Dang, Y., Wu, J.: How task allocation strategy affects team performance: A computational experiment. Journal of Systems Science and Systems Engineering 27, 665–676 (2018)
29. Wang, Z., Nip, K.: Bin packing under linear constraints. Journal of Combinatorial Optimization 34(4), 1198–1209 (2017)

30. Y., C., Lu, L., Yu, X., Li, X.: Adaptive method for packet loss types in iot: An naive bayes distinguisher. Electronics 8(2), 134 (2019)
31. Yu, T., Sekar, V., Seshan, S., Agarwal, Y., Xu, C.: 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. pp. 1–7 (2015)
32. Zhang, B., Dou, C., Yue, D., Zhang, Z., Zhang, T.: A packet loss-dependent event-triggered cyber-physical cooperative control strategy for islanded microgrid. Trans Cybern 51(1), 267–282 (2021 Jan)

Amar Halilovic is currently working as a research assistant and Ph.D. student in Explainable AI and Robotics at the Institute of Artificial Intelligence, Ulm University, Ulm, Germany. He graduated from the Faculty of Electrical Engineering, Department of Automatic Control and Electronics, University of Sarajevo, Bosnia and Herzegovina in 2018 with a BS Degree and in 2020 with an MS Degree. Additionally, he holds an MS degree in Computer Science with specialization in Intelligent Embedded Systems from the Mälardalen University, Västerås, Sweden in 2020. The areas he is most interested in are AI, Robotics, Computer Vision, and Embedded Systems.

Nedim Zaimovic is currently working as an embedded systems engineer in the rolling stock industry. He graduated from the Faculty of Electrical Engineering, Department of Automation Control and Electronics, University of Sarajevo, Bosnia and Hercegovina in 2019 with a B.S degree. In addition to his B.S degree in Electrical Engineering, he holds a M.S. degree in Computer Science with specialization in Intelligent Embedded Systems from the Mälardalen University, Västerås, Sweden in 2020. The areas he is most interested in are control systems, real-time embedded systems, and machine learning.

Tiberiu Seceleanu is professor of Distributed Automation Systems at the Mälardalen University (MDU), Västerås, Sweden, since January 2020. He received his M.Sc. (1994) and Lic.Sc. (1995) degrees from the Polytechnical University in Bucharest, Romania, and the Dr.Tech degree (2001) from Åbo Akademi in Turku, Finland. He became a Docent at Univeristy of Turku, in 2007 in Computer Engineering Systems. A principal Scientist at ABB Corporate, Research (2007-2019) he became a docent at MDU in 2009. Current interests relate to dynamic reconfigurable systems (including networking), high levels of system design, hardware and software, machine learning and autonomous systems.

Hamid Reza Feyzmahdavian received the B.Sc. and M.Sc. degrees in electrical engineering with specialization in automatic control from Sharif University of Technology, Tehran, Iran, in 2005 and 2008, respectively, and the Ph.D. degree in automatic control from the KTH Royal Institute of Technology, Stockholm, Sweden, in January 2016. He is a Principal Scientist with the Control and Optimization Group, ABB Corporate Research Center, Västerås, Sweden. From 2016 to 2017, he was a Postdoctoral Researcher with the Department of Automatic Control, KTH Royal Institute of Technology. His current research interests include distributed optimization and machine learning.

Received: October 01, 2021; Accepted: September 01, 2022.