UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

MASTER'S DEGREE PROGRAMME

Automation Engineering

Graph traversal-based fault management for medium voltage DC shipboard power systems

SUBMITTED AS A MASTER THESIS

to obtain the academic degree of

Master of Science in Engineering (MSc)

by

Christoph Diendorfer B.Sc.

Juli 2016

Supervised by

FH-Prof. Jean D. Hallewell Haslwanter MSc BS

Dr. Karl Schoder

Mark Stanovich PhD

ACKNOWLEDGMENTS

First of all, I would like to thank my research advisor FH-Prof. Jean D. Hallewell Haslwanter MSc BS for guiding me through the various challenges I faced in the course of my thesis. I also would like to thank Dr. Michal Steurer for allowing me a chance to work on the project and being a part of the research team at Center for Advance Power System (CAPS) at Florida State University and for his guidance and encouragement.

I would like to thanks Michael Sloderbeck, Harsha Ravindra, Mark Stanovich and Karl Schoder for their support, advice and suggestions that helped me in my thesis and also patiently discussion about my work.

Last and the most important, I would like to thank my family for believing in me.

UNIVERSITY
OF APPLIED SCIENCES
UPPER AUSTRIA

Wels campus

I hereby declare that I prepared this work independently and without help from third parties, that I did not use sources other than the ones referenced and that I have in-dicated passages ta-ken from those sources. This thesis was not previously submitted in identical or similar form to any other examination board, nor was it published.

Diendorfer Christoph
....
Diendorfer Christoph B.Sc.

Wels, 31.08.2016

Abstract

The Electric Ship Research and Development Consortium (ESRDC) and U.S. Navy are interest of developing the next generation of “all-electric” ships. Part of the goal is to develop a Medium Voltage Direct Current (MVDC) shipboard power system architecture that meets the increasing power and energy demands and have advantages over traditional AC power systems with respect to power density and power distribution efficiency. The MVDC shipboard power systems are designed for fault situations. If a fault occurs in a MVDC shipboard power system it is possible to isolate the faults. With the development of new power electronics the goal is to isolate a fault in less than 8 ms. Therefore, the Fault Management (FM) system is proposed.

To achieve the 8 ms the FM has to perform automatically the isolation in a MVDC shipboard power system. This research presents the steps of the isolation and the transformation of the MVDC shipboard power system to a graph. Using the graph approach the automated rule generation for the identification and detection is described. After a fault was identified the automated isolation sequence is performed by computing an isolation set.
The interaction between the fault detection and location (CFL) system, the electrical network and the FM is also described in this research. Therefore, two communication architecture for the FM were developed and tested. A controller-hardware-in-the-loop (CHIL) testbed demonstrates the two communication architectures.

The results of CHIL demonstrate two FM architectures to perform the isolation of a fault in a medium voltage direct current shipboard power system. The models and analysis in this thesis are validated by experimental results.

Contents

1. INTRODUCTION

1.1. Motivation

The Energy and Power management (P&E) systems are envisioned to automate the processes involved, support higher-efficient use of resources, and aid in fault-handling capabilities. The Center for Advanced Power System (CAPS) is a multidisciplinary research center to perform basic and applied research to advance the field of power systems technology. CAPS has the computational hardware infrastructure to perform real-time simulation of both electric power systems and communication networks. In addition to the dedicated digital real-time simulator (DRTS), embedded controls and computers are available to support Controller Hardware-in-the-Loop (CHIL) testbeds. Several shipboard power system (SPS) architectures are available and have been implemented on the Real Time Digital Simulator (RTDS, [1]), and can serve as a baseline for implementing test cases. The Electric Ship Research and Development Consortium (ESRDC) and U.S. Navy are interest in these shipboard power systems for developing the next generation of “all-electric” ships. Part of the goal is to develop a Medium Voltage Direct Current (MVDC) shipboard power system architecture that meets the increasing power and energy demands [2] and have advantages over traditional AC power systems [3] with respect to power density and power distribution efficiency. MVDC shipboard power systems have been explored for the benefits of flexibility and controlability but not in terms of automated fault management and fault recovery.
The automated fault management and fault recovery question leads to a multidisciplinary research to merge the *electrical world* and the *computer science world* to solve this problem. This work deals with a first step solution for this problem.

1.2. Scope of the work

This section describes the objectives and scope of this work. The objectives of this work are:

- Recovery of a Medium Voltage DC (MVDC) system in less than 8 ms
- Recovery process is fully automated

A Medium Voltage DC (MVDC) shipboard power system (SPS) is designed that faulted sections can be isolated, which considers that no current can flow to the fault anymore. The steps to isolate the fault is loosely defined as the recovery sequence. To achieve the objectives a new automation layer is developed that is called Fault Management (FM). The FM has to communicate with the shipboard power system components and the Centralized Fault Location (CFL) and identification system [4, 5] to perform the recovery sequence. To achieve the 8 ms goal, the communication architecture between the FM, CFL and SPS is discussed in this work. The CFL is a system to monitor the SPS if a fault occurs and after a fault occurs the CFL sends the information to the FM. After receiving the fault information the FM performs the recovery sequence. To automate the recovery steps the electrical grid will be represented as a graph. With the graph representation, adapted graph traversal algorithm can be used to perform the recovery sequence. Also the automatic code generation for the CFL, based on Xing's dissertation [4] and Tamaskar's thesis [5] is possible. The FM layer was demonstrated in a Controller Hardware in the Loop (CHIL) environment. The demonstrator combines real time and power system simulation with external, embedded controls and network platforms. The developed approach and implemented solutions will support demonstrating alternate fault management concepts and be adaptable to future developments.

1.3. Literature survey

A Fault Management (FM) system, based on coordinated interaction between the SPS, a centralized fault identification and location (CFL) system[5] and communication system was *developed* for an efficient recovery for a SPS after a fault occurs.
Tamaskars and Xing's [4, 5] works are dealing with the fault identification and

location of faults in a MVDC system.
Based on the their works the steps for a fully automated recovery was developed, which is described in this work.
The charakteristics of the simulated electrical components of an SPS with the RTDS are described in [6, 7, 8][6, 7, 8].
A representation of an electrical network in a graph was described in [9, 10, 11, 12].

1.4. Thesis Statement and Contribution

The thesis supports the follow thesis statement:

> *Graph based traversal is useful for automating the detection and isolation of faults in a MVDC shipboard power system.*

The thesis should answer the questions:

- How a graph traversal can automate the fault detection in a MVDC SPS
- Is there a way to automate the recovery sequence

With the restriction to recover the MVDC SPS in less than 8 ms a communication architecture between the SPS components and a FM layer is discussed in this work. The design of the communication architecture is also discussed in this work and will answer the computation of the recovery sequence takes place

- Where the computation of the recovery sequence takes place
- Which messages have to be sent between different components
- How the messages can be sentä

1.5. Thesis Outline

This thesis is divided into nine chapters. Chapter 2 describes the notional MVDC SPS and theory about protection schemes. It also describes the fault recovery steps for a MVDC SPS. The next chapter, Chapter 3, outlines the basic definition and notation of the graph theory. It also describes search algorithm based on the graph notation. Based on Chapter 2 and 3, Chapter 4 shows the combination of the *electrical world* with the *graph world* and the algorithms for the fault identification and isolation are described. Chapter 5 deals where the computation of the algorithms from Chapter 4 takes place and the communication architecture between MVDC SPS and external controller. Chapter 6 presents the result of selected case studies. Chapters 7, 8 and 9 are discussing the work, describes the future work and draw conclusions from the research.

2. NOTIONAL ELECTRIC SHIPBOARD POWER SYSTEM

This chapter gives an overview of the electrical components in a medium voltage direct current (MVDC) shipboard power system (SPS). The MVDC architecture as proposed by Norbert Doerry [13] provides an underlying concept for possible implementations of the SPS. A notional version of Doerry's architecture is depicted in Figure 2.1. The goal is to convey the background information of a fault recovery sequence and how to identify faults in the SPS. Doerry's SPS is desigend that if a fault occurs the faulted section can be isolated. The recovery sequence depends on the MVDC SPS and the used components.

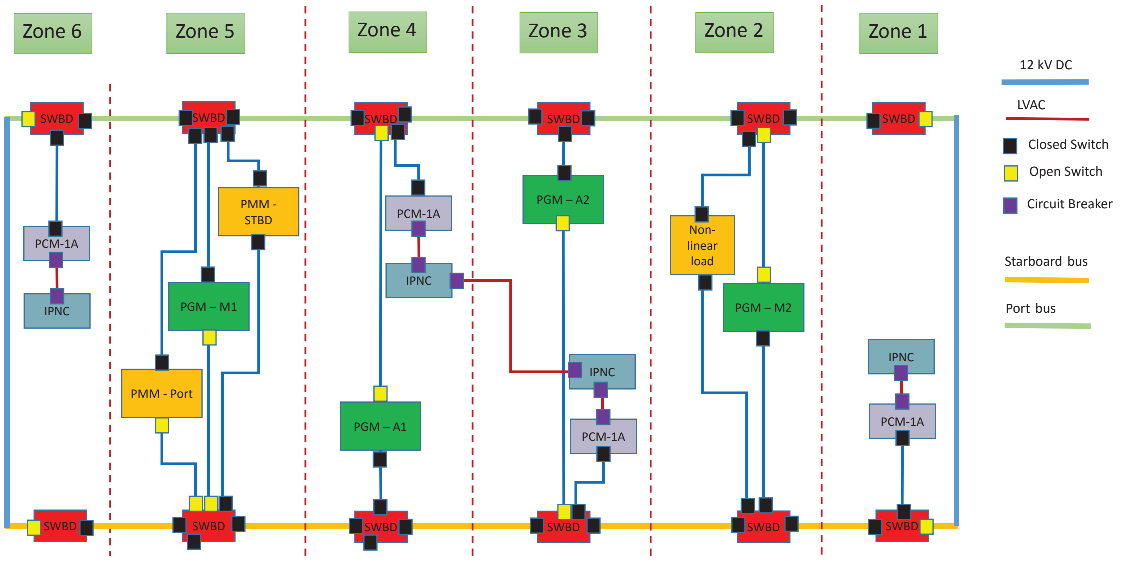


Figure 2.1.: Example: Typical MVDC system configuration

Doerry's proposed architecture is based on a zonal concept [14]. A zone is loosely defined as a subsection of the SPS that is supplied from the MVDC buses. In

Doerry's proposed architecture the model is divided into six zones, which are connected via star-board and port buses. Though the topology would facilitate ring bus operation, it is not a valid option because a single fault would bring down the whole system.
The notional system as despicted in Figure 2.1 has two main Power Generation Modules (PGM-M1 and PGM-M2) and two Auxiliary Power Generation Modules (PGM-A1 and PGM-A2). A PGM contains a Alternating current(AC) power generator (ACPG) and a converter. The converter rectifies the AC current and feeds the MVDC zonal system. The converter of the PGM module is capable of controlling the DC voltage and able to ramp down to zero, if it is necessary. In the example, as shown in Figure 2.1, PGM-M1 and PGM-A2 supply the port

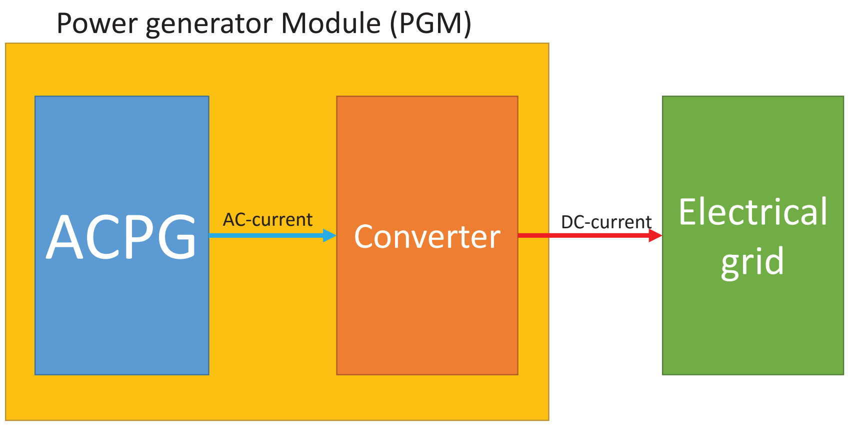


Figure 2.2.: Example: A simplified notional PGM

bus and PGM-M2 and PGM-A1 supply the power for the starboard bus. All electrical components can be connected to either bus, but are getting power from only one side at any given time. The electrical components can be physically disconnected via disconnected switches (DS). The DS are represented in Figure 2.1 with black and orange boxes. A black box means that the DS is closed and an orange box means that the status of the DS is open. The DS can only operate if no current is flowing, otherwise an arc can occur and destroy the DS. There are two ways to isolate faults in a DC network: using circuit breakers or using DS. A circuit breaker can be opened and closed under current. The disadvantages of circuit breakers are, that circuit breaker are heavy, slow and normally not used for frequent operations. Because of these disadvantages DS are used, even with the disadvantage that operating is only able if no current is flowing.
The other components are, but they are not explained in more details in this

work:

- Power Conversion Module (PCM-1A, PCM-1B, PCM-SP);
- Propulsion Motor Module (PMM) ;
- Integrated Power Node Center (IPNC);
- Non-linear load and
- 60 Hz AC distribution system (red line).

The two-bus design has been chosen to support reconfiguration and continued operation in case of faults within zones or on one of the buses. Zonal loads can in these conditions be supplied from the alternate side.

2.1. MVDC Fault Management

In this section the function of a Fault Management system (FM) are explained. In the USA a 60 Hz AC network is used and a cycle needs approximately 16 ms. CAPS set as a goal to achieve a fault recovery in less than a half cycle time. In this work, the electrical system is working with a DC network, which has no cycle. The 8 ms for recovery is a target value that both challenges technologies lines and limits the impacts of fault on the AC generator. Figure 2.3 illustrates the recovery sequence if a fault occurs in a MVDC network.[15]

The fault recovery sequence consists of four steps:

1. Detect and locate fault

   A fault identification system monitors currents of the electrical system and if a fault occurs and identifies fault locations

2. De-energize system

   All generator sources, which supply power to fault section, must stop energize the SPS.

3. Isolate fault

   After the FM has de-energized the system, it can start to isolating the faulty subsystem. Because all major sections components connect via a DS, it is possible to isolate fault sections by opening DS.

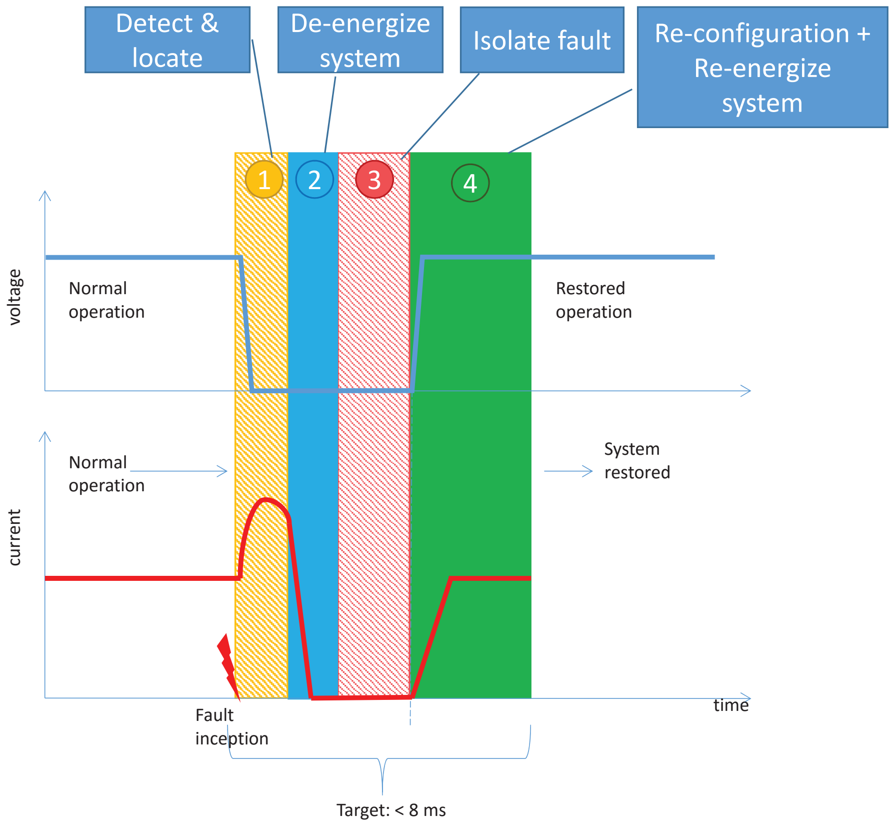


Figure 2.3.: Recovery sequence in MVDC network

4. Re-configurate

   With the information of the fault location, the system topology and component characteristics, it is possible to re-configure system power to the remaining operating system. The reconfiguration may consider prioirites and vital function aspects to make best use of resources.

5. Re-energize system

   Soureces can be restored and controllable feeds activated.

How to re-configure a system after a fault and to re-energize will not be discussed in this work. But for for the implementation a hardcoded solution was done.
Figure 2.3 shows typical voltage and current traces observed during a fault recovery sequence. The moment a fault occurs, current increases rapidly. The controls of the PGM tries to bring back the system to the current limited mode [16] to protect components. After the fault is identified, the system must be de-energized in order to open the DS. After the faulted system is isolated, the remaining sys-

tem can be re-energized. For example, Figure 2.4 illustrates a fault in Zone 2. A fault at one of the cables causes PGM-M2 to lose the capability to supply energy through the starboard bus. The system can be reconfigured to allow PGM-M2 to supply loads through the port bus by opening DS3/ DS4 and closing DS1/ DS2.

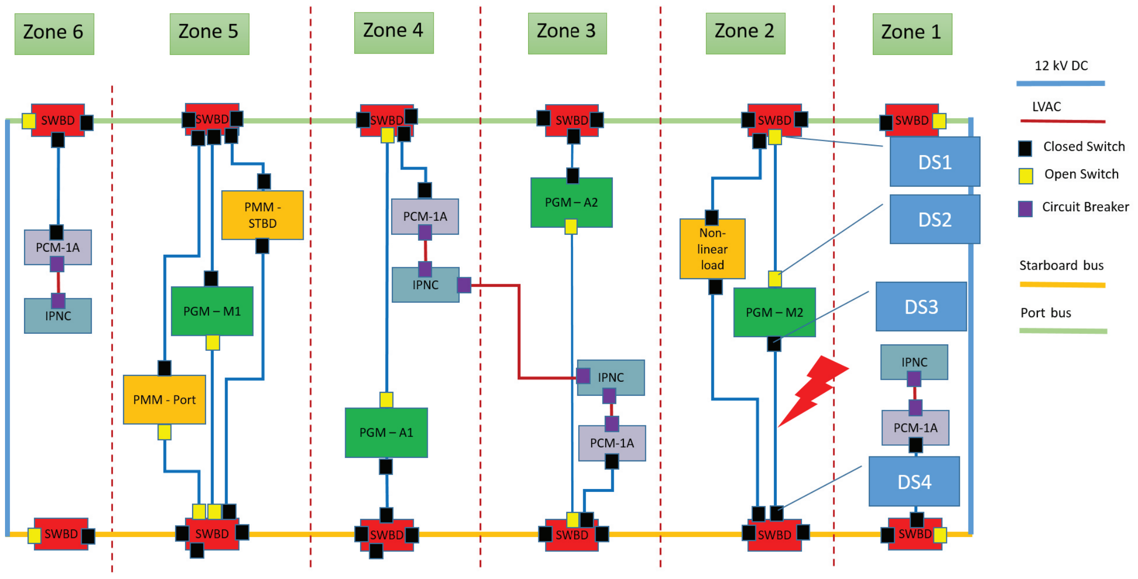


Figure 2.4.: Example: section failure before isolation and reconfiguration

2.2. Protection scheme

In this section, a protection scheme and a slightly adapted version of it [17, 18, 19, 5] will be explained. The protection scheme is based on Kirchhoff's Current Law (KCL) and satisfied the requirements of high adaptability and speed for the location identification approach [4].

2.2.1. Percentage Differential Protection

The Percentage Differential Protection PDP scheme is based on the idea of differential current relay [17]. PDP defines two types of currents:

- Operating current I_{OP} and
- Restraint current I_R

Figure 2.5 illustrates and example cable section with assigned current measured to illustrate the PDP Principle.

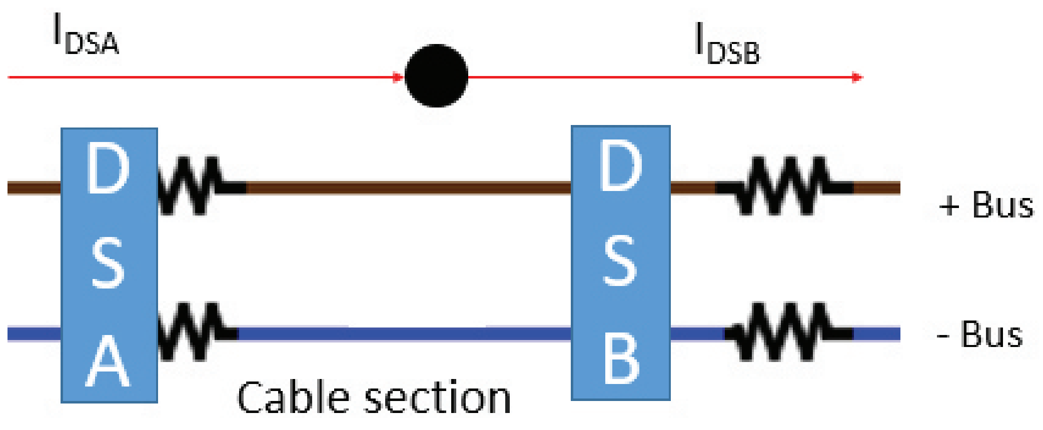


Figure 2.5.: A cable section with two current measurments

I$_{\mathrm{OP}}$ for the shown setion is calculated as follows:

$$I_{OP} = |I_{DSA} - I_{DSB}| \tag{2.1}$$

Without loss of generality, currents going into the protection section are assumed to be positive and all leaving currents are negative. I$_{\mathrm{OP}}$ can be compute as follows:

$$I_{OP} = |\sum_{i=1}^{n} I_i| \tag{2.2}$$

In case of a fault I$_{\mathrm{OP}}$ will be greater than zero. An example is shown in Figure 2.6. Both currents in the supervised section, I$_{\mathrm{DSB}}$ and I$_{\mathrm{DSA}}$ will flow to the fault location, which may be a short circuit in the bus.

The literature and relay manufacture typically calculate restraint current I$_{\mathrm{R}}$ as follows[19]:

$$I_R = k\,|I_1 + I_2 + ... + I_n| \tag{2.3}$$

$$I_R = k\,(|I_1| + |I_2| + ... + |I_n|) \tag{2.4}$$

$$I_R = max(|I_1|, |I_2|, ..., |I_n|) \tag{2.5}$$

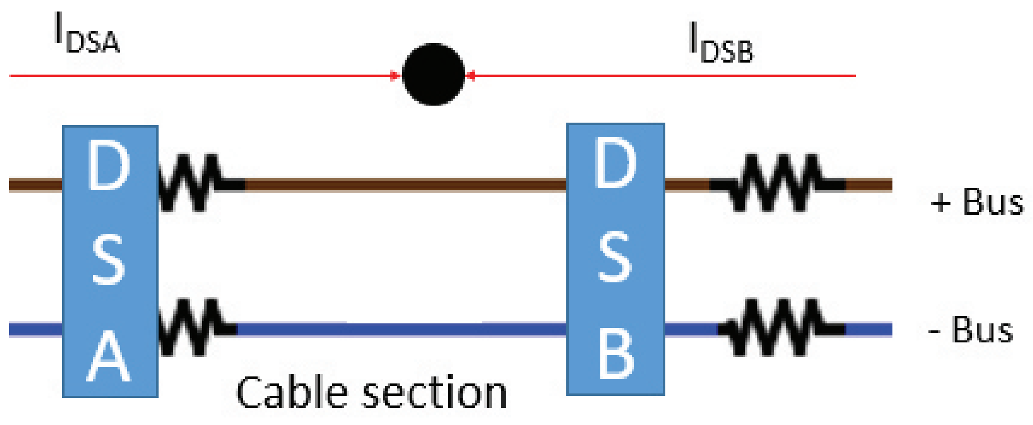


Figure 2.6.: Example: Current flow during a fault

- k is a scaling factor
- I_1, ... ,I_n: instantaneous currents at monitored section

In this research the restraint current is calculated as follows:

$$I_R = |I_{DSA}| + |I_{DSB}| \tag{2.6}$$

In the following conditional equation, the PDP identifies a fault if the condition[1] is true.

$$fault = \begin{cases} true, & if: \frac{I_{OP}}{I_R} > Slope \\ false, & otherwise \end{cases} \tag{2.7}$$

with:

- Slope, which is a property chosen constant value

The slope signifies the sensitivity of the PDP to the current. The slope characteristic in Figure 2.7 provides high sensitivity when low levels of current are flowing in the zone of protection but has less sensitivity when high levels of current are flowing [19].

[1]

$$indicator = \begin{cases} true, & if: condition \\ false, & otherwise \end{cases}$$

2.2.2. Adapted Percentage Differential Protection

A slightly adapted version of the PDP is called Adapted Percentage Differential Protection (APDP). In the literature on APDP different equations are used [19]. In this section, the conditional equations are shown but not explained in more detail. The conditional equation defines, if there is a fault or not.
The first one is from Xing's dissertation[4] and the equation is as follows:

$$fault = \begin{cases} true, & if: |\sum_{i=1}^{n} I_i| - K \sum_{i=1}^{n} |I_i| \geq I_{Min} \\ false, & otherwise \end{cases} \tag{2.8}$$

with n is the total number of currents in the section:

- $\mathrm{I_i}$: instantaneous current values
- K: restraint coefficient
- $\mathrm{I_{min}}$: minimum operating current value

Miao, Liu, and Lin's paper[18] describes APDP with the following equations:

$$fault = \begin{cases} true, & if: |I_1 + I_2| > I_{Min}\, and\, |I_1 + I_2| > K|I_1 - I_2| \\ false, & otherwise \end{cases} \tag{2.9}$$

with:

- K: restraint coefficient
- $\mathrm{I_1}$, $\mathrm{I_2}$ are current values on protected line

Miao, Liu, and Lin's paper[18] following Figure 2.7 was created

Another relationship is mentioned in Tamaskar's master thesis[5]., with the APDP condition was described as:

$$fault = \begin{cases} true, & if: \frac{|\sum_{i=1}^{n} I_i|}{\sum_{i=1}^{n} |I_i| + I_{Min}} \geq Slope \\ false, & otherwise \end{cases} \tag{2.10}$$

with n is the total number of currents in the section:

- $\mathrm{I_{min}}$: minimum operating current value

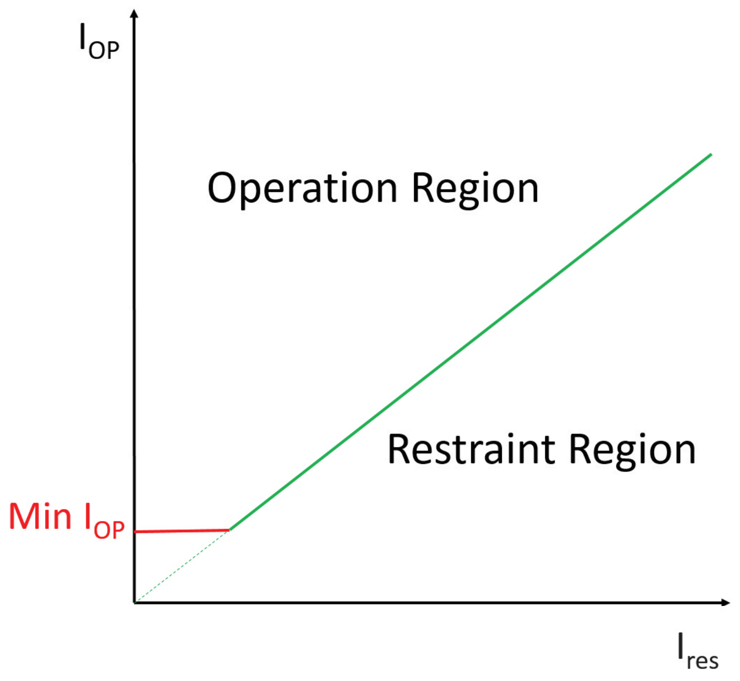


Figure 2.7.: The adapted percentage differential protection diagram

- I_i: instantaneous current values
- S: the slope coefficient

2.3. Conclusion

This chapter described a novel medium voltage DC shipboard power system and its components. The SPS components have different tasks during a fault sequence. The fault sequence consits of 5 steps; Detect and locate fault, De-energize system, Isolate fault, Re-configurate and Re-energize system. To detect a fault in the medium voltage DC system differential protection schemes are explained. With the knowledge of the steps for the fault sequence and the electrical components, an automtic fault recovery was realized..

3. GRAPH NOTATION AND DEFINTION

"A graph consists of a finite set of vertices, a finite set of edges, and a rule which tells us which edges join which pairs of vertices." Biggs [20]

The purpose of this chapter is twofold: introduce (1) the basic definitions of graph theory and (2) the associated representation of graphs. These fundamental blocks provide the basics for achieving the goal of this work which is to automate the detection and isolation process in an MVDC system.
Therefore, the chapter starts with the explanation of the Big O Notation. The Big O notation gives an idea about the computational effort of algorithms, which will be used for to evaluate the further developed algorithms for the recovery. After the Big O section was described, the graph notation and definition are described.

3.1. Big O Notation

To understand computational effort of an algorithm and compare algorithms, the big O ($\mathcal{O}$, pronounced "big-oh") notation is used. This notation shows the worst-case runing time of an algorithm on inputs of size n. With the information of n and the running time it is possible to calculate a function f(n). This function becomes the upper bound of the running time of the algorithm.
The $\mathcal{O}$-notation is defined as follows: [21]

$$\begin{aligned}\mathcal{O}(g(n)) = \{f(n) :\ & \textit{there exists positive constants } c \textit{ and } n_0 \textit{ such that} \\ & 0 \leq f(n) \leq cg(n)\ \textit{for all } n \geq n_0\}\end{aligned} \tag{3.1}$$

with a given function $g(n)$ and :

- n: integer value
- c: constant value
- n_0: minimum possible value

To calculate the complexity of an algorithm, the easiest way is to look at the pseudocode. Pseudocode of an Insertion-Sort algorithm is shown in Algorithm 3.1. Each line of the pseudocode needs an execution time Δ_i. With the assumption that comments are not executable statements, they will have no computation time. The number t_j is the count time of the while-statement in line number 4 for the value i. The execution times and execution frequency, are in Table 3.1.

Algorithm 3.1 Insertion_Sort(A)

```
Input: A: Array with random integers
1: foreach j = 2 to length [A ] do
2:     insert_key = A[j]
   ▷ Insert A[j] into the sorted sequence [1 . . j - 1].
3:     i = j - 1
4:     while i > 0 and A[i] > insert_key: do
5:         A[i + 1] = A[i]
6:         i = i - 1
7:     end while
8:     A[i + 1] = insert_key
9: end foreach
```

Table 3.1.: Execution count and time for the Insertion-Sort algorithm

PSEUDOCODE	Count	EXECUTION TIME
for j <- 2 to length[A]	n	Δ_1
do insert_key <- A[j]	n-1	Δ_2
i <- j - 1	n-1	Δ_3
while i > 0 and A[i] > insert_key	$\sum_{j=2}^{n} t_j$	Δ_4
do A[i + 1] <- A[i]	$\sum_{j=2}^{n}(t_j - 1)$	Δ_5
i <- i - 1	$\sum_{j=2}^{n}(t_j - 1)$	Δ_6
A[i + 1] <- insert_key	n-1	Δ_7

With the information of Table 3.1 it is possible to calculate the runtime. The runtime for the Insertion-Sort algorithm is:

$$\begin{aligned} T(n) &= \Delta_1 n + \Delta_2(n-1) + \Delta_3(n-1) + \Delta_4 \sum_{j=2}^{n} t_j + \\ &\quad \Delta_5 \sum_{j=2}^{n}(t_j - 1) + \Delta_6 \sum_{j=2}^{n}(t_j - 1) + \Delta_7(n-1) \end{aligned} \tag{3.2}$$

To calculate the function g(n) the worst-case has to be determined. If the input array is reverse sorted, the worst case has happened. In this case, each element in the entire subarray has to be compared and the while loop execution count can be calculated as follows:

$$\sum_{j=2}^{n}(t_j) = \frac{n(n+1)}{2} - 1 \tag{3.3}$$

$$\sum_{j=2}^{n}(t_j - 1) = \frac{n(n-1)}{2} \tag{3.4}$$

With (3.3) and (3.4) the worst-case runtime can be calculated as follows:[1]

$$\begin{aligned} T(n) &= (\frac{\Delta_4 + \Delta_5 + \Delta_6}{2}) * n^2 + (\Delta_1 + \Delta_2 + \Delta_3 + \frac{\Delta_4 - \Delta_5 - \Delta_6}{2} + \Delta_7) * n \\ &\quad -(\Delta_2 + \Delta_3 + \Delta_4 + \Delta_7) \end{aligned} \tag{3.5}$$

A notional version of (3.5) is given in (3.6).

$$T(n) = un^2 + vn + w \tag{3.6}$$

With the assumption that T(n) only takes nonnegative values and T(n) is the worst-case runtime function the following equation can be used:

$$T(n) \leq cg(n) \tag{3.7}$$

$g(n)$ is the highest power of $T(n)$ and c is a constant value.

Equation (3.7) indicates the connection between the runtime function T(n) and the Big O notation. The big O expresses the upper bound of the growth rate and not the exact growth rate of the function[22]. For the example (Algorithm 3.1) the bound is given by $\mathcal{O}(n^2)$.

[1]see Cormans [21] Appendix A for a review of how to solve these summations

3.2. Graph

This sections describes the defintion of a graph G and the different types of graphs. A graph G was defined by Lovász and Vesztergombi [23]as:

$$G = (V, E) \tag{3.8}$$

where:

- V: Set of verticies
- E: Set of edges

Where an edge e is a pair of vertices and can be described as follows:

$$e = (v_1, v_2) \tag{3.9}$$

where:

- $v_1 \,\varepsilon\, V$ and
- $v_2 \,\varepsilon\, V$

Graphs can be directed or undirected and Figure 3.1 illustrates an example for each type. In a directed graph, each edges is an ordered pair of vertices (set), which means the edge has an orientation. In an undirected graph, an edge is an unordered pair (set) of vertices and the edge has no orientation.

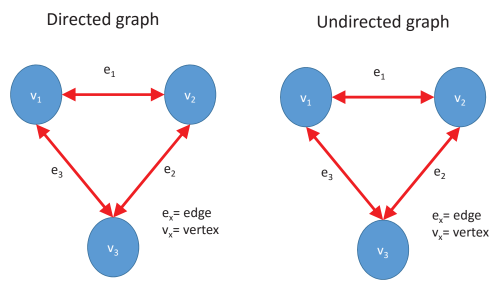


Figure 3.1.: Example for comparing directed and undirected graphs.

3.3. Graph Data Structure

Data structures are used to store a graph for use by a computer program. The different types of data structures will be described in this section. In this thesis an undirected graphs are most relevant and therefore, the data structures will be explained for them.
In the literature different data structures are mentioned, these are edge list, adjacency lists and adjacency matrices. These data structures differ in memory requirements and how long it takes to do certain opterations on the graph. These differences will be explained with the example as shown in Figure 3.1. Two criteria will be used to compare the different representations:

1. Case: memory or space requirements to store edges
2. Case: determination time whether a given edge is in the graph

With the big O notation the two mentioned criteria will be compared. At first the edge list will be explained.

3.3.1. Edge list

A simple way to represent a graph is a list, or array, of edges (E), which is called edge list. For an edge list, the total space is $\mathcal{O}(|E|)$ 1 because each edge contains two vertices. To determine if an edge is in the edge list, an algorithm has to search through the edge list. The time will increase linearly with the length of the edge list and so the big O notation is $\mathcal{O}(|E|)2$. The example from Figure 3.1, the graph G can be represented as edge list in a computer program as is shown below.

$$G = \{\, (v_1, v_2),\ (v_2, v_3),\ (v_1, v_3)\,\}$$

3.3.2. Adjacency matrices

A graph with |V| vertices has an adjacency matrix with the size of |V| x |V|. This matrix contains only 0 and 1. The entries (i, j) (row i and column j) of the matrix are 1 if the edge (i, j) is in the graph. The adjacency matrix is called A. To find out if an edge a(i,j) is in the graph, the entry for g[i][j] must be 1. If the entry is one, the edge exists.

$$a_{i,j} = \begin{cases} 1 & if:\ (i,j) \in E, \\ 0 & otherwise. \end{cases} \tag{3.10}$$

The memory usage of an adjacency matrix is $\mathcal{O}(V^2)$ for saving the full matrix. To determine if a edge e(x,y) exists only the entry (x,y) has to be checked and the computational effort is $\mathcal{O}(1)$
For an undirected graph, the adjacency matrix is symmetric: the row j, column j entry is 1 if the row j, column j entry is 1. For a directed graph, the adjacency matrix need not be symmetric.

Graph G, for the example from Figure 3.1, can be represented as a adjacency matrix in a computer program as it is shown below.

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

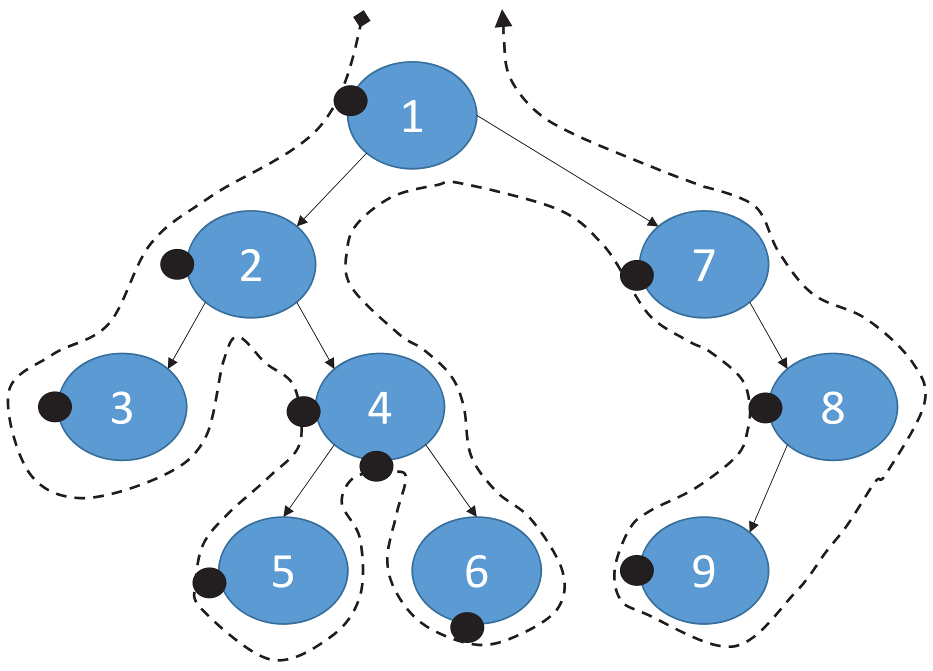
3.3.3. Adjacency list

This type of representation is a combination of the edge list and the adjacency matrix. For each vertex, the adjacency list will be stored with the neighbors. To determine if an edge e(i,j) is in the list, an algorithm goes to i's adjacency list and checks if it contains j. This computation is $\mathcal{O}(d)$, d is the degree of vertex i, because each element in the list must be compared. The degree of vertix i can be $|V|$-1 if i is adjacent to all other $|V|$-1, or it can be 0 if it is an isolated vertex.|
If an edge $e = (u, v)$ is an undirected edge then u appears in v's adjacency list and vice versa. Because of that, the sum of the lengths of all adjacency lists is $\mathcal{O}(2|E|)$, and represented the memory usage of $\mathcal{O}(|E|)$ [21]. An example to represent graph G as an adjacency list is shown above.

$$\begin{aligned} G \ = \ [\, & v_1 = [v_2, v_3], \\ & v_2 = [v_1, v_3], \\ & v_3 = [v_1, v_2]] \end{aligned}$$

3.4. Graph Search

This section describes an application of using graph representation. In computer science, graph search or graph traversal is a process of checking/ visiting or updating each vertex in a graph. These traversals are classified by the order in which the verticies are visited. During a graph traversal it may require that some vertices be visited more than once and therefore, these verticies getting *marked* as vistited. To mark the vertices should prevent from trasveral the graph indefinitely. If a vertex was already visited, it is ignored, otherwise the vertex will be checked and updated as visited.
Different versions of graph traversal are described in literature: Pre-Order traversal, In-Order traveral and Post-order traversal[24]. Figure 3.2 ilustrates an example for a graph and the solution for the mentioned traversal operations.



Pre-order: 1-2-3-4-5-6-7-8-9
In-order: 3-2-5-4-6-1-7-8-9
Post-order: 3-5-6-4-2-9-8-7-1

Figure 3.2.: Solution for travseral operation

Two common applications based on the graph traversal are Depth-first search (DFS), Breadth-first search (BFS).

3.4.1. Depth-first search

The first version of a DFS was investigated in the 19th century by a French mathematician Charles Pierre Trémaux. Charles Pierre Trémaux tried to find a strategy for maze problems.[25] . A recursive implementation of DFS is shown in Algorithm 3.2[21]. The Algorithm 3.2 begins the graph traversal with a given vertex and explores along a single path into the depth, before backtracking.

Algorithm 3.2 DFS(G, v)

```
Input: Graph G and v as start vertex
1: mark v as visited
2: foreach unvisited n in neighbors(v) do
3:     DFS(G,n)
4: end foreach
```

In computer science the DFS is mostly used to traversal the whole graph. To traversal the entire graph, it can happend that vertices are visited more than once and consequently the traversal takes time $\mathcal{O}(|V| + |E|)$[21].

3.4.2. Breadth-first search

The BFS was first developed by E.F.Moore in the late 1950s to find the shortest path out of a maze [26] and discovered independently as a wire routing algorithm by C.Y.Lee [27, 28]. The BFS algorithm takes an arbitrary vertex of the graph and explores the neighbors first, before moving into the depth. A recursiv implementation of BFS is shown in Algorithm 3.3. The time complexity is $\mathcal{O}(|V| + |E|)$[21], because in the worst case every vertex and edge will be visited.

3.5. Conclusion

This chapter described the graph notation and definition. At first the big O notation is described for the comparison of the different graph representations and the computational effort of graph traversal algorithm. The graph representations

Algorithm 3.3 BFS(G,v)

```
Input: Graph G and start vertex
1: foreach n of neighbors(v) do
2:     if (vertex n is not visited) then
3:         mark n as visited
4:     end if
5:     foreach n in neighbors(x) do
6:         BFS(G,x)
7:     end foreach
8: end foreach
```

are compared in fact of their usage of memory space and the computational power for finding an edge e. Table 3.2 shows the summary of the computational effort of the different graph representations.

Table 3.2.: Summarization of the computational effort of the different graph representations

	1. Case	2. Case
Edge list	$\mathcal{O}(\|E\|)$	$\mathcal{O}(\|E\|)$
Adjacency matrices	$\mathcal{O}(\|V^2\|)$	$\mathcal{O}(1)$
Adjacency list	$\mathcal{O}(\|E\|)$	$\mathcal{O}(d)$

Based on the fact of the computational effort each graph representation has its benefits. The computational effort for the DFS and BFS is the same. Depending on the task either DFS or BFS has to be chosen.

4. GRAPH BASED IDENTIFICATION AND ISOLATION ALGORITHMS

This chapter describes a way to automate two steps of the recovery sequence, the fault identification and the fault isolation, based on a graph approach. Therefore, an electrical topology is stored as a graph G. The transformation of an electrical shipboard power system into a graph is described in this chapter. The graph abstraction is used for autotmatic rule generation for the fault identifcation and performing the fault isolation.

4.1. Transformation

A given connectivity matrix of an electrical grid [29] is used to generate a graph. Connectivity matrix indicates whether pairs of electrical devices are connected or not in the electrical network. With the information of the given connectivity matrix M_c the graph G is created. Therefore, at first an arbitrary device d_i is picked and added as a vertex to the graph. The next step is to look for all connected devices which are electrically coupled in the connectivity matrix. If devices d_i and d_c are connected an edge e(d_i ,d_c) is added to the graph. Eacch device has electrical charakterisitcs which are added to the new created graph vertex x. These characteristics are:

- Is a source: $True, False$
- Can isolate : $True, False$
- Can conduct current: $True, False$
- Actual Status: $True, False$

These information are used to automate the fault isolation. Refered to Chapter 2 the steps for isolation a fault are described. To isolate the fault each device which can isolate the fault has to be found. Also the sources which are suppling power to the fault has to found. Therefore, these attributes have to be saved with the new generated vertex x. Algorithm 4.2 shows the pseudo-code for adding the attributes to the vertex.
For example, a PGM is a source, which can change the current value in the system, can isolate a fault and conduct current. If the second and third attributes are $True$ the device will be a DS for example. Other information can maybe use for the re-configuration, but is not discussed in this work.

Algorithm 4.1 add_attribute(device)

```
Input: device
Output: device with attribute
 1: if (device is Source:) then
 2:     device.Source = True
 3: end if
 4: if (device is Isolate:) then
 5:     device.Isolate = True
 6: end if
 7: if (device is Can_Conducte_Current:) then
 8:     device.Can_Conducte_Current = True
 9: end if
10: if (device Status:) then
11:     device.Status = True
12: end if
13: return vertex
```

From a given connectivity matrix M_c all devices are extracted and saved in a device list. This list will be used for Algorithm 4.2 to generate the graph G .

An example connectivity matrix M_c is shown in Table 4.1.

Table 4.1.: A given connectivity matrix 4.3

	Source1	Source2	Load	DS1	DS2	DS3	Bus
Source1	0	0	0	1	0	0	0
Source2	0	0	0	0	1	0	0
Load	0	0	0	0	0	1	0
DS1	1	0	0	0	0	0	1
DS2	0	1	0	0	0	0	1
DS3	0	0	1	0	0	0	1
Bus	0	0	0	1	1	1	0

Algorithm 4.2 transformation(device_list)

Input: device_list: contains all devices from electrical topolgy
Output: graph G

```
 1: Graph G = Null
 2: foreach d in device_list do
 3:     G.add_vertex(d)
 4:     remove d from device_list
 5:     foreach neighbor x of d do
 6:         G.add_vertex(x)
 7:         G.add_attributes(x)
 8:         G.add_edge(d,x)
 9:     end foreach
10: end foreach
11: return G
```

An example is shown in Figure 4.1. At first a device list is created with the devices: Source 1, Source 2, Load, DS1, DS2, DS3 and Bus. As example Source 1 is taken and will be represent as a vertex. The attributes Source, Isolate and Can_Conduct_Current are added as shown in Algorithm 4.1. The connected device d_c is DS1, therefore an edge e(Source, DS1) (represented as dotted line) is added to the new generating graph G.

With the given connectivity matrix M_c (from Table 4.1) the corresponding graph G looks like as shown in Figure 4.2.

4.2. Generating Detection and Identification rules

A SPS is designed that if a fault occurs it should be possible to isolate the fault and use the rest of the electrical topology. Power-systems protection is a branch of electrical power engineering which deals with the protection of electrical power systems. The goal of the protection is to keep the power system operationel stable by isolating only components that are under fault and leaving as much of the power system as possible in functional operation. Therefore, the SPS can be partioned by Cable Section (CS) to isolate faults if necessary. These CS are monitored by the CFL [4, 5] to identifies a fault if one occurs. The system calculates, one of the previous mentioned protection scheme for every CS.
If a fault occurs and is detected, the fault location is added to the graph G as a

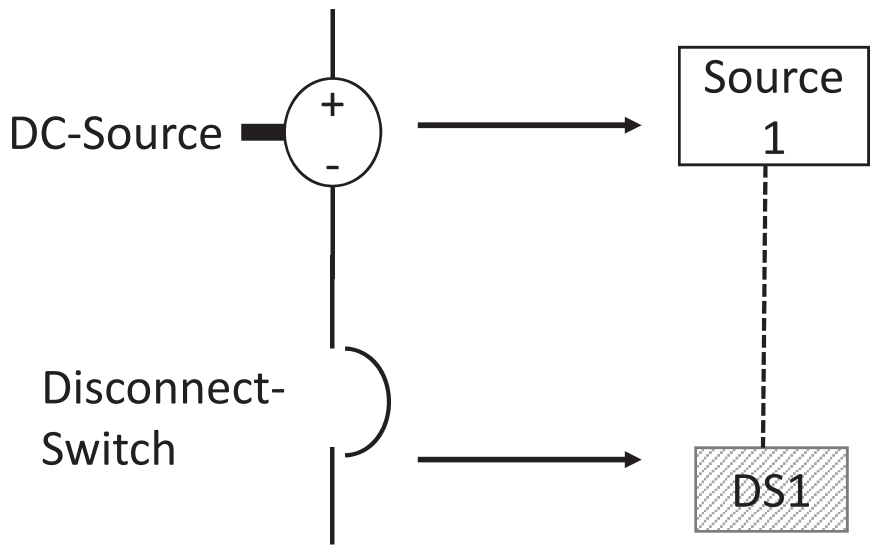


Figure 4.1.: Electrical device represented as vertex in Graph

new vertex. This new vertex is called fault vertex and is defined as:

Definition 1. A fault vertex, is a vertex representing the fault location of the electrical grid in the graph and has the attribute to conduct current.

To generate all possibilities, where fault can occur the graph G can be splited into subgraphs. If a fault vertex is added to the graph G, it is possible to find the subgraph containing the fault vertex. These subgraphs are called sections and defined as:

Definition 2. An isolationable section S are connected vertices (subgraph) of G. A isolation section S can be electrically isolated from other components of the electrical network.

$$S \subseteq G \tag{4.1}$$

The graph based fault identification partitions the graph so that each vertices is monitored. If a fault occurs in one of the sections the CFL should identify. The computation of the section can be done offline and is called identification rule and is defined as:

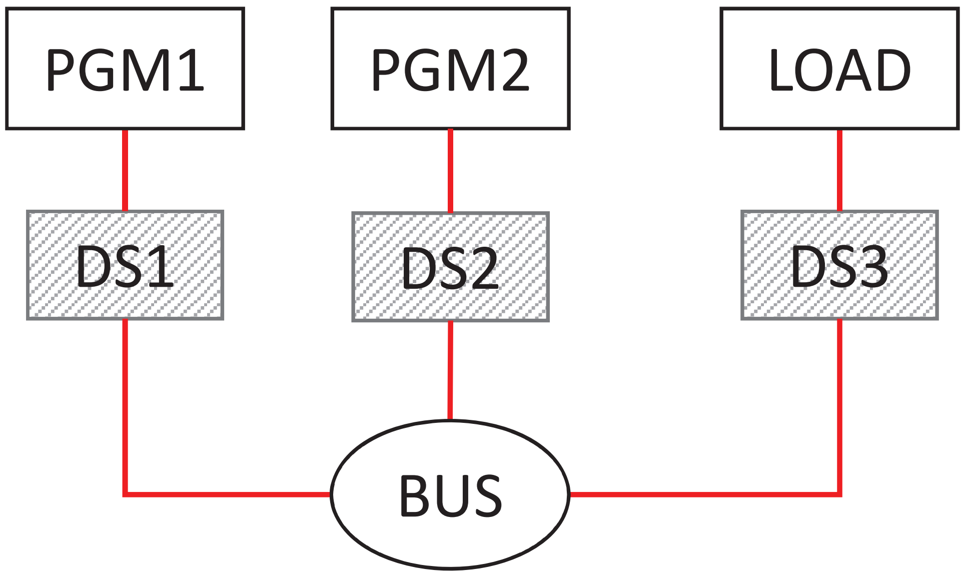


Figure 4.2.: Given connectivity Matrix represented as graph G

Definition 3. An identification rule rule is a sequence of instructions to detect a fault by using the PDP scheme.

To each fault an unique identification number (ID) is added. This ID contains the information of the location and the time. Therefore, the Universally Unique Identifier (UUID) can be used. This step is not discussed in further details in this work.

Algorithm 4.3 gives a pseudocode for detecting a fault in the electrical network.

Algorithm 4.3 fault_exits(R)

```
Input: Graph G
1: S = min_subgraph(R)
2: foreach s in S do
3:     if (Calculation of PDP in s) then
4:         s contains fault_vertex
5:     else
6:         continue
7:     end if
8: end foreach
9: create unique ID
10: return fault_vertex and unique ID
```

4.3. Fault Isolation Rules

Here is described the process of isolating a fault in an electrical network based on graph traversal algorithm. The status attribute as mentioned before contains the information of the actual status of the electrical device. For example, if a DS is open, which means no current is flowing at this DS, the status attribute will be $True$. This information is needed to get the feedback of the electrical devices, during the fault sequence.

Definition 4. An fault isolation rule is a sequence of instructions to isolate a fault vertex, depending on the online configuration of the electrical topology

Figure 2.3 shows the recovery sequence.
When a fault occurs in the electrical network the recovery sequence should be perfomed automatically. At first the involved components for the isolation have to be computed. The involved components for the recovery are:

- all electrical connected devices to the fault location, which can change the current level in the system
  - for example: all kind of sources (power generator with converter, battery, ...)
- isolation devices

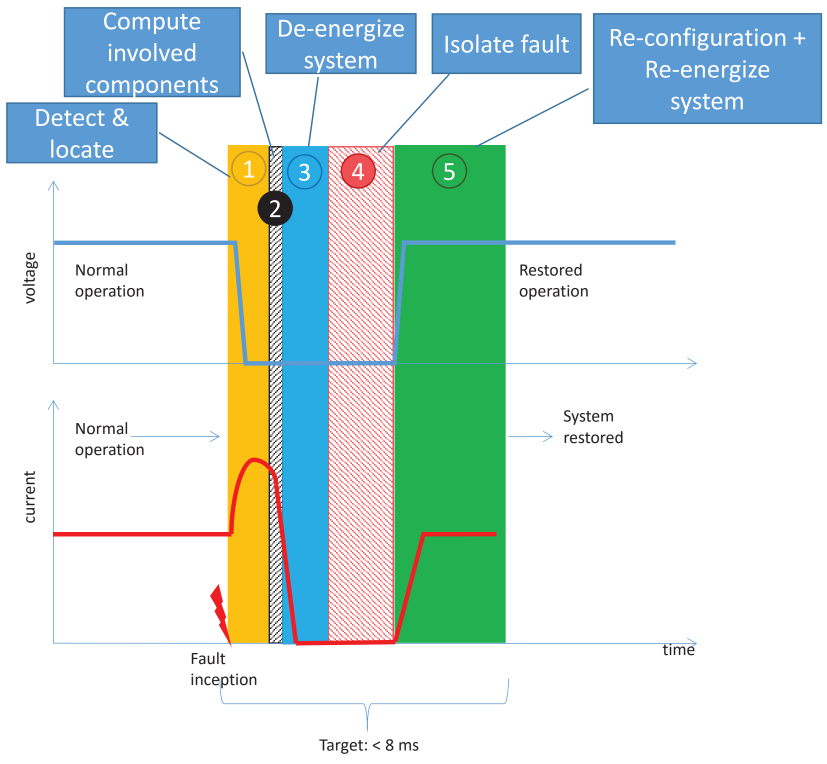
Figure 4.3 illustrates the extende recovery sequence. It shows, that after a fault occurs the involved components are calculated and these devices are used for the recovery sequence.

After a fault vertex is added to the graph G the involved components can be found by graph traversal. Therefore, algorithms were developed which are based on the previous basic graph traversal algorithms. An isolation set is defined as:

Definition 5. An isolation set, is a subset of devices from the isolation section S with the attribute to isolate a fault..

To evaluate all isolation sets ι, Algorithm 4.4 is used. It uses the actual and entire graph and starts with the fault vertex to traversal the graph. If an isolation device is found, it is added to the isolation set ι.

After the isolation set ι from the isolation section S is found all paths from the vertices to the vertices with the attribute "is a source" have to be found. Thus a



d

Figure 4.3.: New recovery sequence

DFS can be used for this goal, and will compute for every vertex in the isolation set ι the path to a vertex which has the mentioned attribute. Algorithm 4.5 shows a pseudocode to use DFS for this purpose.

4.4. Conclusion

In this chapter the *union* between the *electrical world* and the *graph theory* was described. The goal was to take the information of the connectivity matrix of a Shipboard Power System (SPS) and represent it with a graph and add attributes to the vertices. Each vertex represents an electrical device of the SPS and an edge the cable between the electrical devices.
After the transformation of the SPS to the graph G, rules for the identification and isolation of faults are developed. These rules are based on graph traversal algorthim and allows automated fault identifcation and isolation of faults.

Algorithm 4.4 isolation_devices(G,v)

Input: Graph G and vertex v
Output: a set of devices needed for isolation

```
 1: isolation_set =
 2: mark v as visited
 3: if (v can conduct current) then
 4:     if (v can isolate) then
 5:         add v to isolation_set
 6:         return isolation_set
 7:     end if
 8:     foreach unvisited n in neighbors(v) do
 9:         rec_value = isolation_devices(G,n)
10:         add rec_value to isolation_set
11:     end foreach
12: end if
13: return isolation_set
```

Algorithm 4.5 find_reachable_sources(G,v)

Input: Graph G and v is an isolating device
Output: set of devices

```
 1: isolation_set =[]
 2: if (v can conduct current) then
 3:     if (v can isolate) then
 4:         add v to isolation_set
 5:         return isolation_set
 6:     end if
 7:     foreach unvisited n of neighbors(v) do
 8:         rec_value = find_reachable_sources(G,n)
 9:         add rec_value to isolation_set
10:     end foreach
11: end if
12: return isolation_set
```

5. FAULT MANAGEMENT ARCHITECTURE

In this chapter a new developed Fault Management (FM) system will be introduced. The goal with the new FM system is, to automate the recovery sequence in less than 8ms. Therefore, the algorithms from Chapter 4 are used. The new FM layer takes place between the Power and Energy Management (P&E) and the Centralized Fault Location and identification system (CFL) . A shipboad power system design contains following layer: Power & Energy Management (P&E), the new FM, CFL and the electrical network. Figure 5.1 illustrates the layers and communication between the layers. The FM layer communicates with the CFL, the electrical components, P&E and issues the commands for automated recovery process. The communication between P&E and FM is not discussed in this work.

The FM implementation is based on the previous discusssed graph recovery algorithms. The FM handles also the communication with the different layers.

5.1. Fault Recovery Sequence performed with FM based on graph approach

In this section the Fault Recovery Sequence (FRS) is described. After a fault is identified the FM performs the Algorithm 7.1 and computes all involved components. The invlolved components are the isolation set ι and now the shipoard power system has to be de-energized. Therefore, the FM sends the propiate commandos to the certain devices as shown in Algorithm 5.1.

After the system is de-energized the isolation device, which only can isolate if no current is flowing, can change their status as it is shown in Algorithm5.2

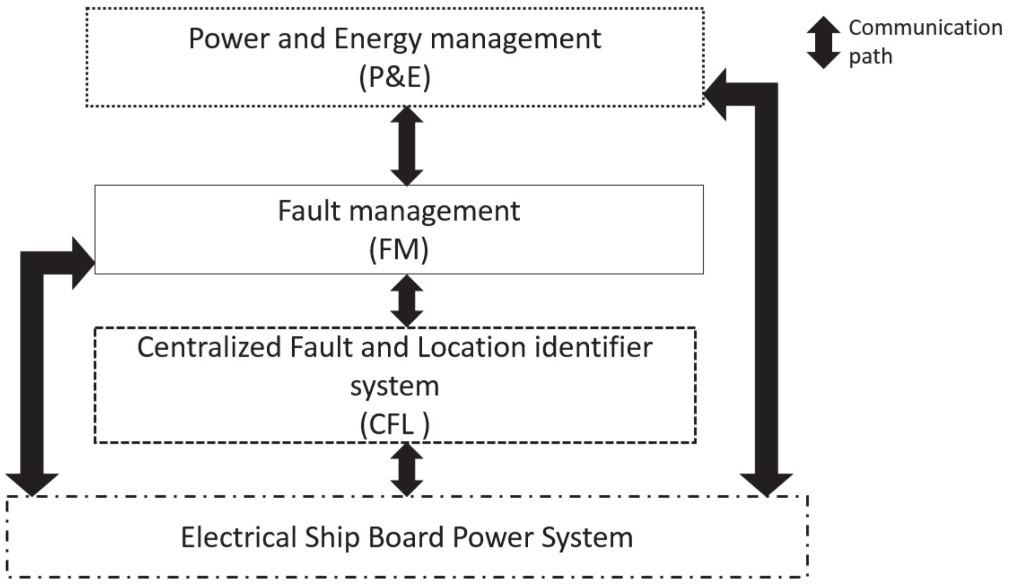

Communication path
Power and Energy management (P&E)
Fault management (FM)
Centralized Fault and Location identifier system (CFL)
Electrical Ship Board Power System

Figure 5.1.: Illustration of the communication between the layers

Algorithm 5.1 de_energize(Isolation_set)

Input: Isolation_set: pre computed with previous mentioned algorithm
Output: system is de-energized

```
1: foreach d in Isolation_set do
2:     if (d is source) then
3:         send_message(d,ramp_down)
4:     end if
5: end foreach
6: return True
```

As already mentioned, re-configuration and re-energize is out of the scope of this work.

5.2. Data Communication patterns

This section describes two data communication pattern. These pattern are used for the implementation of the communication between the electrical network, CFL and FM (Figure 5.1).

The goal of data communication is to prepare rules and regulation that computers with different operating systems, languages and location can share information. The rules and regulations are called protocols and standards. Three fundamental

Algorithm 5.2 isolate(Isolation_set)

Input: Isolation_set: pre-computed with mentioned algorithms
Output: fault is isolated

```
1: foreach d in Isolation_set do
2:     if (d can isolate and d not source) then
3:         send_message(d,open)
4:     end if
5: end foreach
6: return True
```

characteristics are important to describe the efficiency of a data communication network:

- **Delivery**: In the telecommunication shared data have to receive to the correct destination.
- **Accuracy**: The delivered data must be exact in context of information
- **Timeliness**: Late delivered data

There are different ways, in which the communication can take place. In this work two communication established patterns will be used and these are called Publisher/ Subscriber (Pub/Sub) and Request/ Reply(Req/Rep) pattern.

5.2.1. Publisher/Subscriber pattern

The Publisher/ Subscriber (Pub/Sub) system was described the first time 1987 [30]. This pattern is used in architectures [31], where senders of messages, called publishers, are not sending messages to a specific receiver, called subscribers. They publish the messages into the *world*[1] and subscribers chooses the receive messages as needed.

Message filtering In a Pub/Sub model, subscribers receive a subset of all messages. The mechanism of selecting the messages for processing is called *filtering*. Two common ways are mentioned in literature for *filtering* the messages; topic-based and content based.
In **topic-based** system, messages contain a specific topic. All subscribers will receive all publishes messages and check after that if the topic is interesting for

[1]Here it means a network topology with a number of subscribers.

them or not. The publisher has to define the topic of the message frame. A simplified message frame contains the topic and the information. All subscribers will receive the message, but only the subscribers, which are listen to the specific topic will use this information.
In a **content-based** system, publisher delivers messages only to a subscriber if the attributes or content of those messages match with the constraints defined by the subscriber. The subscriber is classifying the messages.

5.2.2. Request/ Reply pattern

The Request/ Reply(Req/Rep) pattern is a very common communication method. For example, it is used browsing on th world wide web. Someone, who opens a web page and starts the request to get information is called requester. The web page which replies is called replier. This pattern is also valid applied for a computer to computer communication and is very often the basis of client-server architecture[32].
In Req/Rep pattern the requester sends a message to a specific replier in a system. This replier sends a message back with an information. This simple pattern allows a two-way conversation with one another. The conversation can be hold synchronously over a channel and asynchronously and is often refereed to “synch over asynch” or “synch/async”. Figure 5.2 illustrates an example of a Rep/Req pattern for a server-client architecture. The client sends the request: “What is your actual status?”. After the server receives the message, it sends back it actual status. In the illustrated example, the server sends back the information: “I am busy”.

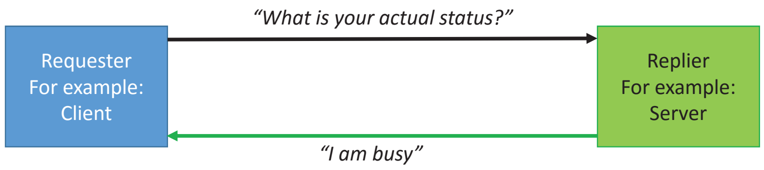


Figure 5.2.: Server Client example using the Rep/Req pattern

5.3. Central vs Distributed Fault Management

This sections describes different possibilities where the computation of the automated fault recovery algorithm can take place and how the communication between the different layers will happen and compare them. Depending on the number of hardware where the computation takes place the FM will be called as Centralized Fault Management (CFM) or a Distributed Fault Management (DFM). To compare these two architectures, an architecture in this work is defined as:

Definition 6. An architecture is a communication design between fault management, centralized fault and location identifier and electrical components.

To compare architectures the following assumption are made:

- Each FM knows the whole topology of the electrical grid, which is represented as graph
- All electrical devices have own controller with main logic and send their status back and react on commands from the FM

5.3.1. Central Fault Management

One way to perform the recovery sequence is to use only one FM instance. With the example as it is shown in Figure 2.1, a CFM can be used as it is demonstrated in Figure 5.3.

The CFM communicates with multiple CFLs and can communicate with each electrical device in the electrical topology. For each zone a CFL is used to monitor the zone. With the assumption that the CFM communicates directly with each electrical device for the communication, the number of messages which can be sent in parallel depends on the number of available ports.

5.3.2. Distributed Fault Management

> “A distributed system is a model in which components located on networked controllers communicate and coordinate their actions by passing messages.” Coulouris, et al. [33]

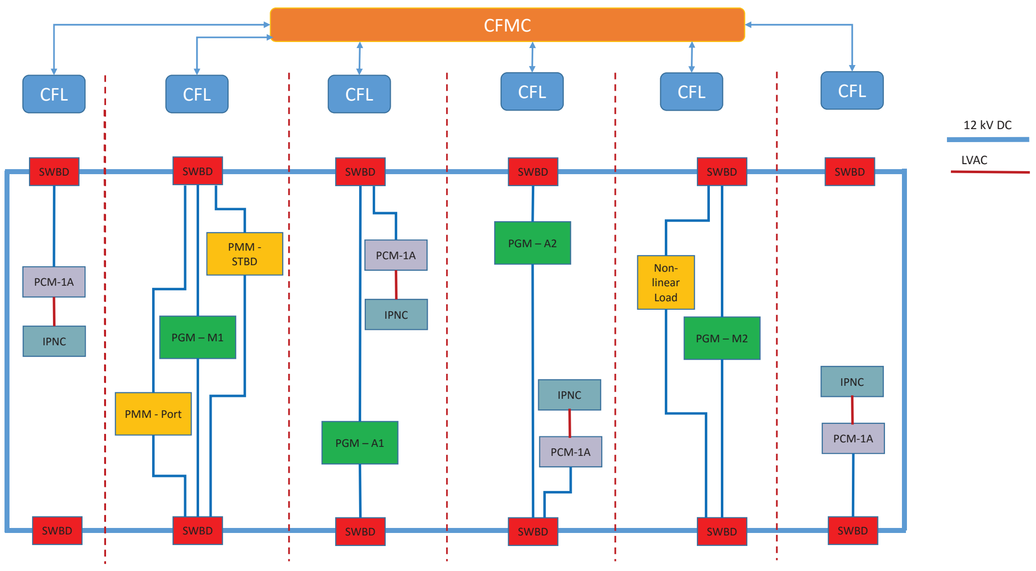


Figure 5.3.: Example of a centralized fault management architecture

This subsection describes the developed distributed architecture for FM system. Figure 5.4 illustrates the developed DFM architecture. In there multiple DFM are used to perform the recovery sequence. To use this architecture, the following assumptions are made:

- Each electrical device has a controller with main control and replies with its status
- Each DFM knows the overall system topology which is stored as graph
- CFLs publish fault location and unique ID

According to the assumptions the developed architecture grants to share computational power and communication ports. In fact, that each control of the distributed architecture computes the same recovery sequence, the tasks can be performed parallel. For example, the one control in the system sends the commands to de-energize the system and another control will send the commands to the isolation devices of the isolation set ι. If a controller fails to compute the isolation set ι or sending the commands, another controller is able to take over and the recovery will still be performed.

In addition to Figure 5.4, each DFM is capable to communicate with each electrical device directly. This communication line is not shown in the Figur1e 5.4.

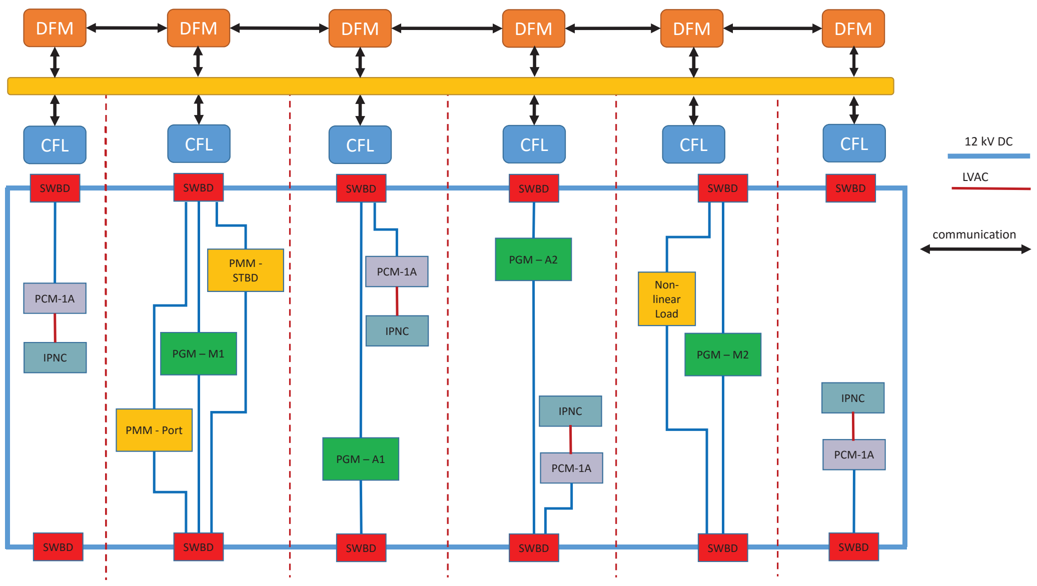


Figure 5.4.: An example for a distributed fault management system

With the distributed architecture it is possible that multiple controllers perform the recovery sequence for a fault. With the assumptions made, it is possible to parallelize the fault recovery sequence. Therefore, Figure 5.5 shows a time sequence diagram for a fault recovery sequence using two DFM. The CFL communicates directly with the components of the electrical network. After a fault occurs, which was identified by the CFL, the CFL publishes the message with the location and a unique identifier. DFM receives the message and start immediately with the fault recovery. Both of them identify the involved components to isolate the fault. FM1 sends the command for ramping down all conducted PGMs and parallel FM2 sends the open commanding for the isolation devices, which can only change their status without current flow. After sending the commands, both FM start to compute the re-configuration and the first FM, which finishes will send the commandos to ramp up the sources. The solution for this step was done hardcoded and will not explained in this work. Both DFMs will compute the same re-configuration, because they use the same algorithm and have the same knowledge about the electrical topology.

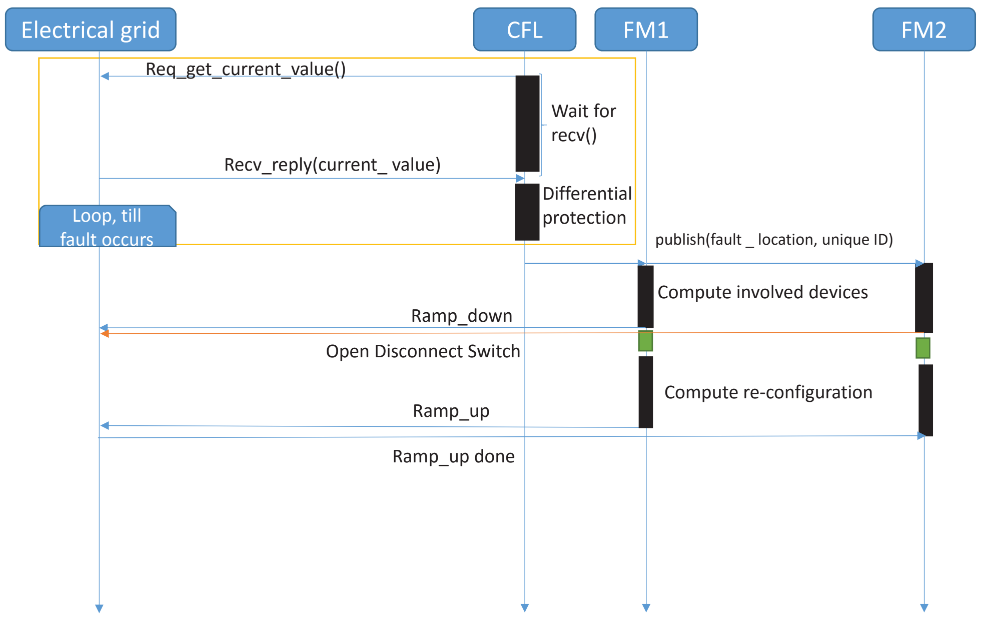


Figure 5.5.: Simplified time sequence diagram fo fault supervision and recovery process

After the fault occurs the DFMs have less than 8ms to perform the recovery steps. With the made assumption it is possible to work in parallel and share communications ports.

5.4. Conclusion

This chapter describes two fault management architectures. At first two common communication patterns are explained, which are used for the FM to communicate with the CFL and the electrical devices of the shipboard power system. For the communication between the FM and the electrical devices the request/reply pattern was used. Between the CFL and FM the publisher/subscriber pattern is used.
After the communication was explained, the computation of the recovery sequence is explained. Therefore, two architectures are developed: the Centralized

Fault Management (CFM) with using one instance of the FM approach or the Distributed Fault Management (DFM) with using multiple instances of the FM approach.
The goal of this chapter was to describe the different architectures and the working principle.

6. METHODS

This chapter describes the used methods for generating data and later to analyze them. The goal of the work is automated fault identification and performing automatically the fault isolation. The fault isolation should be done in less than 8 ms.

Xing [4] developed a Centralized Fault Location and identification system (CFL), which was the basis for Tamaskars thesis [5].. Both of them dealt with the fault identification in a MVDC system. To automate the fault identification the graph approach is used. Based on Algorithm 4.3 a fault can be identified and therefore, a Percentage Differential Protection (PDP) scheme has to be chosen. Equation 2.10 was used for the PDP calculation. In Tamaskars thesis [5] is shown, that a fault in the MVDC shipboard power system can be identified in less than 1 ms. The Equation 2.10 is also used in Tamaskars thesis and was proved by measurements that the Equation 2.10 can identify different kinds of faults in the MVDC shipboard power system.
After the fault is identified the next steps of the recovery sequence are performed by the Fault Management system. The automatic performance of these steps, De-energize, Isolation, Re-configuration and Re-energize was not discussed in literature until now. I choose the graph approach to use well-known graph traversal algorithm to automate the recovery steps.
At first the involved components are computed with the Algorithms 4.4 and 4.5. After the involved components are computed the Algorithms 5.1 and 5.2 are performed by the FM. The computation of the recovery sequence was done with one FM instance to perform the CFM approach and also with 2 FM instances to show the DFM architecture.
With the CFM and DFM approach, communication architectures were developed to perform the recovery steps. These architectures are based on well-known architectures and adapted to achieve the goal of 8 ms. Also the communication pattern between CFL, FM and electrical network was chosen.

To demonstrate the working principle the same electrical network based on Tamaskars [5] was used to simulate different fault locations. The generated rules for the fault identification should be able to identify the faults in the electrical network and later the FM performs the fault recovery sequence. To compare the CFM and DFM approach, for each fault location the current and voltage curves in the electrical network were measured. The goal with these measurements is to show that it is possible to identify and recover the MVDC system in less than 8 ms. In the next chapter the implementation is described in more detail.

7. FAULT MANAGEMENT IMPLEMENTATION

To demonstrate the automated fault recovery sequence in a MVDC network, a test setup was implemented to mimic the approach. This chapter presents the controller hardware in the loop (CHIL) testbed implementation of the FM architectures and the recovery sequence for various fault locations. All recovery sequences tested with the shown electrical network in Figure 7.1, which was developed by Andrus, et.al [8]. The electrical grid is described in this chapter in more details. The chapter also describes the a centralized fault location and identification system, which is a system based on the percentage differential protection scheme.

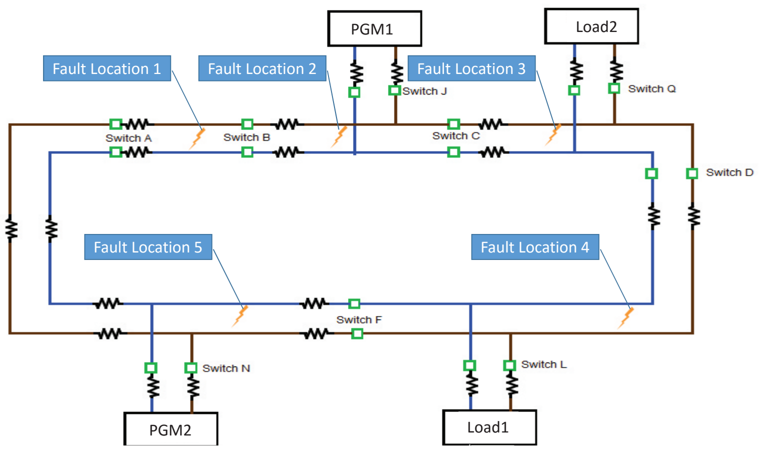


Figure 7.1.: Electrical testbed, which is simulated in real-time with RTDS

7.1. Simulated Electrical network Description

This section describes the configuration of the MVDC network from Figure 7.1. The schematic representation of the MVDC network from Figure 7.1 has the following components:

- 9 Disconnect Swichtes (DS): all of them are closed
- 2 Ppwer Generator Module (PGM)
- 5 fault locations
- 2 loads

As mentioned a PGM contains a power generator and an AC/DC converter. In this experimental setup, Modular Multilevel converters (MMC) are used. The operation principles and MMC topology were originally proposed in [34, 6, 35]. The operation principle of MMC can be categorized as DC/AC converter or AC/DC converter. The typical AC/DC MMC is used for High Voltage DC (HVDC) application, but in this work it is used for a medium voltage DC application. In this work the MMC performs as Voltage Source Converter (VSC) or as Current Source Converter (CSC). The MMC control is aimed at driving the output DC voltage to a previous defined voltage and limiting the current in a fault situation. Previous research work was focused to use DC/AC MMC on using MMC as a motor drive [36, 37, 38, 39, 40]. The electrical shipboard power system is configured as follows:

- PGM1 with MMC operates as VSC
- PGM2 with MMC operates as CSC
- Load1 and Load2 are MMC

The electrical network has five CS which can be isolated if a fault occurs. The CSs and the respective DS are shown in Table 7.1 and in Figure 7.2.

Table 7.1.: Sections and the respective disconnect switches

Section ID	DISCONNECT SWITCH		
1	DSA	DSB	-
2	DSB	DSC	DSJ
3	DSC	DSD	DSQ
4	DSD	DSF	DSL
5	DSF	DSN	DSA

In this work DS are used to isolate the CS where the fault occurs. An new ultra fast disconnect switch was tested at CAPS [15], which is able to open in less then 1 ms.

7.2. Centralized Fault Location and Identification System

The centralized fault location and identification system [4] (CFL) calculates the PDP scheme for each CS to identify the faults in the CS. To calculate the PDP for the CS the current value of the isolation devices is needed. In Tamaskars thesis[5] an approach for a communication architecture for the CFL approach is shown.

The CFL is monitoring multiple sections as shown in Figure 7.2.

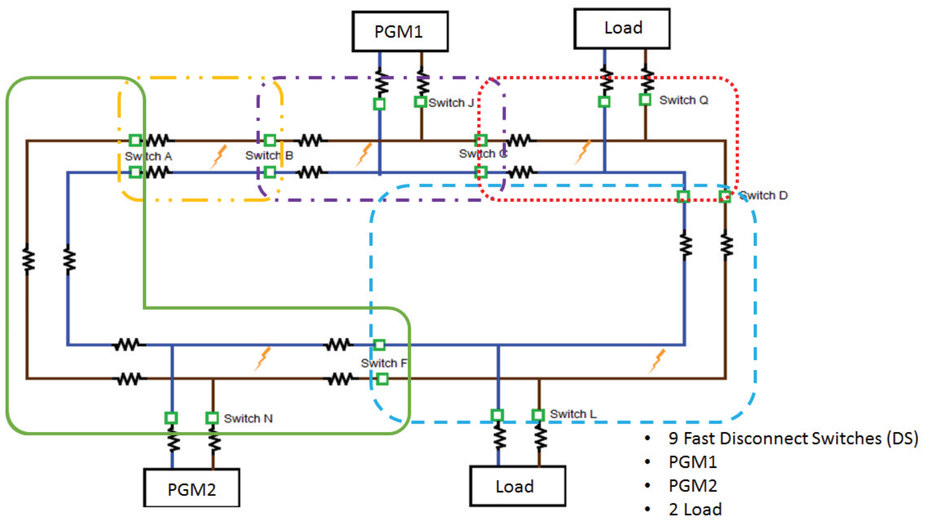


Figure 7.2.: Example: CFL monitored sections

7.2.1. Hardware Specification

In [5, 4] a system with distributed measurements units was discussed. The system collects the current measurements and communicate with a controller which performs a percentage differential protection scheme.

CAPS has demonstrated this control system with equipment from Beckhoff. This industrial-grade automation platform consists of a master and multiple slaves. The master performs the calculations for the protection scheme and the slaves are the distributed measurement units. To communicate between master and slaves, EtherCAT [41] is used.

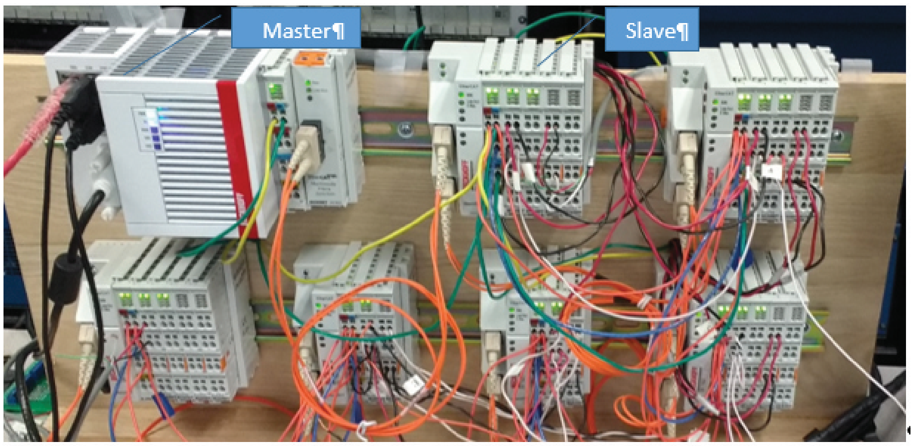


Figure 7.3.: Distributed embedded control: master and slave control units of the Beckhoff platform

The benefits of this system are the support of computation times of a 100 µs and the EtherCAT real-time communication capabilities. Figure 7.4 shows an example of using the CFL with the Beckhoff system. The green boxes are representing the slave components and the orange box is the master.

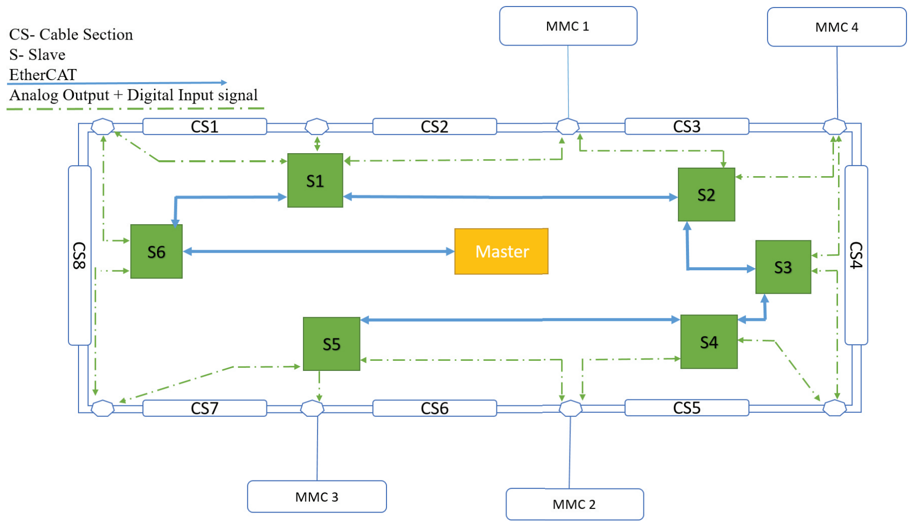


Figure 7.4.: Example MVDC system with supervision by CFL

7.3. CHIL Testbed

The hardware for testing the FM approach is described in this section. The electrical network from Figure 7.1 is simulated with a Real-Time Digital Simulator (RTDS). Real-time simulation means that the computation of the simulated electrical network or system advances one moment in each period of wall-clock time [42].
To communicate with RTDS it is possible to use fibre optic. To evaluate the signal values of the fibre optic connection a GTFPGA Xilinx ML507 board, which has a fibre protocol capability (2Gbps), is used. The GTFPGA is mounted in a DELL OPTIPLEX760 desktop computer with a Intel(R) Core(TM)2 Duo CPU E7600 @ 3.06GHz processor and runs the Ubuntu 14.04 X86 as operating system. Via the PCI-express slot it is possible to exchange data between the GTFPGA and the desktop computer.

Because the communication with the RTDS is done via the GTFPGA board a server-client architecture was developed. The desktop computer is running as a server and provides all necessary function for the clients to exchange data between clients and RTDS. The clients are the FM and CFLs of the system. The server program is based on the python multithreading example [43] and allows multiple requests. The server-client behaves similar to the Beckhoff system, which was mentioned in 7.2.1. The benefit of using this hardware instead of using the Beckhoff was, not to deal with the Beckhoff environment. This saved time for developing the algorithms and the GTFPGA board with the server-client architecture has a similar behaviour like the Beckhoff system.
For the distributed architecture, the FM program was implemented on the Versalogic Mamba (Mamba) boards. The Mamba boards use a 2nd generation Intel Core 2 Duo processor, which is designed for specifically embedded applications and runs up to 2.26 GHz[44]. Debian 8 was used as operation system on the Mamba boards. Figure 7.5 shows the hardware and the communication paths of the CHIL testbed.

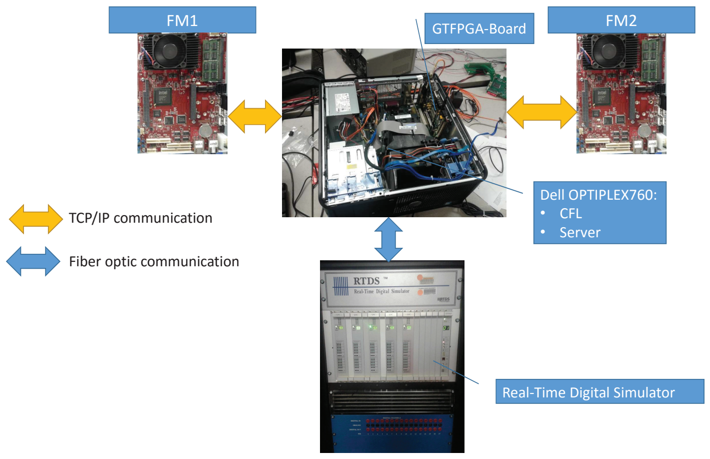


Figure 7.5.: CHIL testbed

7.4. Communication for Distributed Fault Management

This section describes the communication between the CFL, FM and the electrical network. Figure 7.6 illustrates the communication between the different controller based on the CHIL testbed. (Figure 7.5). A layer called boards and the GTFPGA board.

The communication between FM and the server was realized with the request reply pattern. To send a message from the FM to the Server via Ethernet cable the messages is packed into a protocol. In this work the messages was packed into the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol. The TCP/IP protocol specifies how data are packeted, addressed, transmitted routed and received. The TCP/IP protocol guarantee that the messages arrives at the specified address. The conversation between the FM and server were hold synchronously. This grants to send a feedback immediatlly to the FM from server with the information of the request. The disadvantage of this implementation is, if the server is not responding, the FM will not continuing working, until it gets the feedback. Because if the FM sends a request, it will be in a waiting state untill it receives a message.

For the communication between CFLs and Mamba boards the Pub/Sub pattern was used. Therefore, the ZeroMQ library is used, which is a high-performance asynchronous messaging library for distributed systems [45].

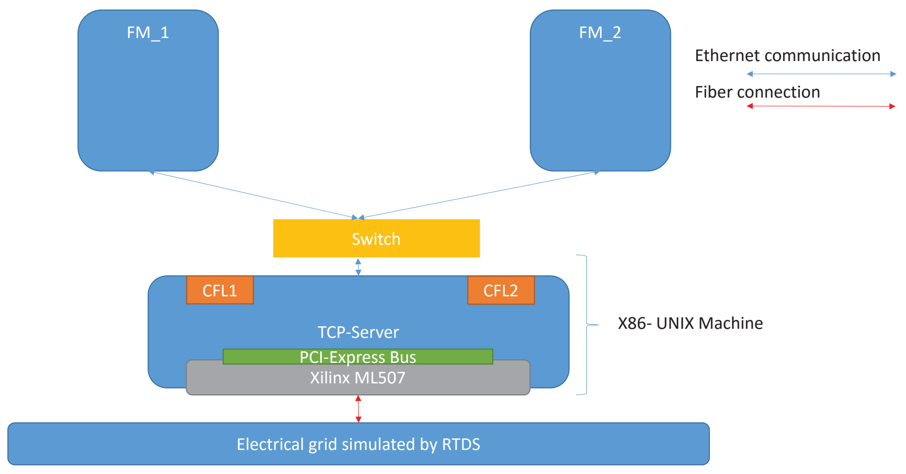


Figure 7.6.: Schematic hardware set up for DFM

The data exchange between the server and the RTDS was done via the PCI-express card to the GTFPGA board and fibre optic.

7.5. Communication for Central Fault Management

To evaluate the CFM approach the same hardware was used. The FM algorithm and CFL algorithm were running on the desktop machine. Because both were running on the same machine, the server was not necessary and therefore, the FM and CFL communicates directly with the GTFPGA board. Figure 7.7 illustrates the CFM communication.

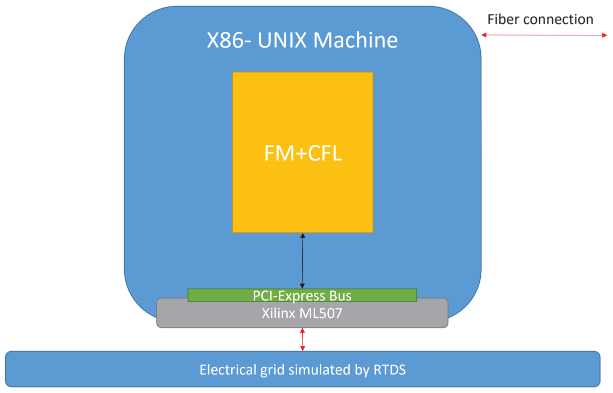


Figure 7.7.: Schematic hardware set up for CFM

7.6. CHIL setup

In this section the CHIL process is described. At first the transformation of the given electrical network (Figure 7.1) into a graph is shown and its representation in the computer program. After that the implementation of the communication between the different hardware is described.

7.6.1. Transformation

As described in Section 4.1 an electrical network can be represented as graph. Therefore, the given connectivity matrix M_c will be used as adjacency matrix. Figure 7.8 shows the connectivity matrix of the proposed electrical grid.

With the given connectivity Matrix M_C the generated graph is shown in Figure 7.9.

For saving memory space the electrical grid in this work was stored as an adjacency list rather than the adjacency matrix. Figure 7.10 shows the python re-presentation of an adjacency list for the given connectivity matrix M_C.

Figure 7.8.: The connectivity matrix M_c for the proposed electrical network

	DA	DB	B1	DJ	P1	DC	B2	DQ	L1	DD	B3	DL	L2	DF	B4	DN	P2
DA	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
DB	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
DJ	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
P1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
DC	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
B2	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0
DQ	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
L1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
DD	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0
B3	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0
DL	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
L2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
DF	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
B4	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
DN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
P2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

7.6.2. Compute the involved components

In this subsection the pseudocode for the computation of the involved components during a fault sequence is shown. Based on the Algorithms 4.4 and 4.5 a merged algorithm was developed. Algorithm 7.1 shows the merged version. For the merging version a copy for the visiting list must be created. This will effect the computation power of the algorithm, which is not explained in this work.

7.6.3. Central Computation

For the CFM approach, the computation for the recovery sequence is performed with one controller. In this work, the recovery sequence program was computed on the desktop machine. Figure 7.11 shows the sequence diagram for the CFM approach, after a fault occurs. For this sequence it was assumed that only one

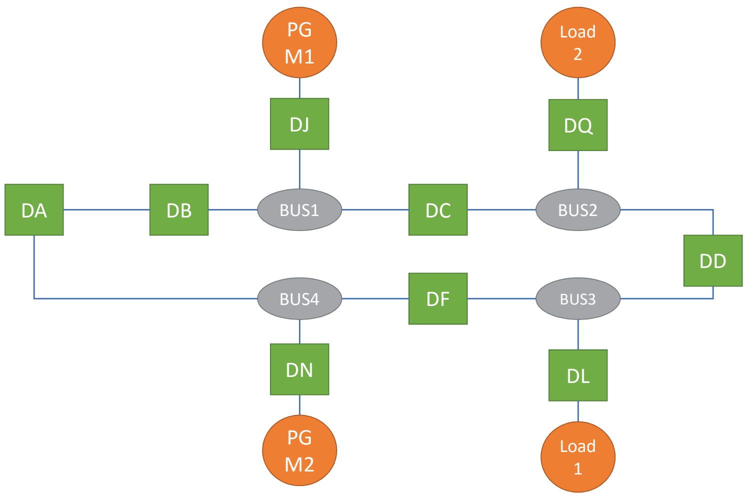


Figure 7.9.: Electrical grid is represented as graph

CFL was used to monitor the whole system.

```
g = {   DA  : [DB,B4],
   DB  : [DA,B1],
   B1 : [DB,DC,DJ],
   DJ  : [B1,P1],
   P1 : [DJ],
   DC  : [B1,B2],
   B2 : [DC,DQ,DD],
   DQ  : [B2, L2],
   L2: [DQ],
   DD  : [B2, B3],
   B3 : [DD,DL,DF],
   DL  : [B3, L1],
   L1: [DL],
   DF  : [B3, B4],
   B4 : [DA,DF,DN],
   DN  : [B4, P2],
   P2: [DN],}
```

Figure 7.10.: Connectivity matrix represented as adjacency list

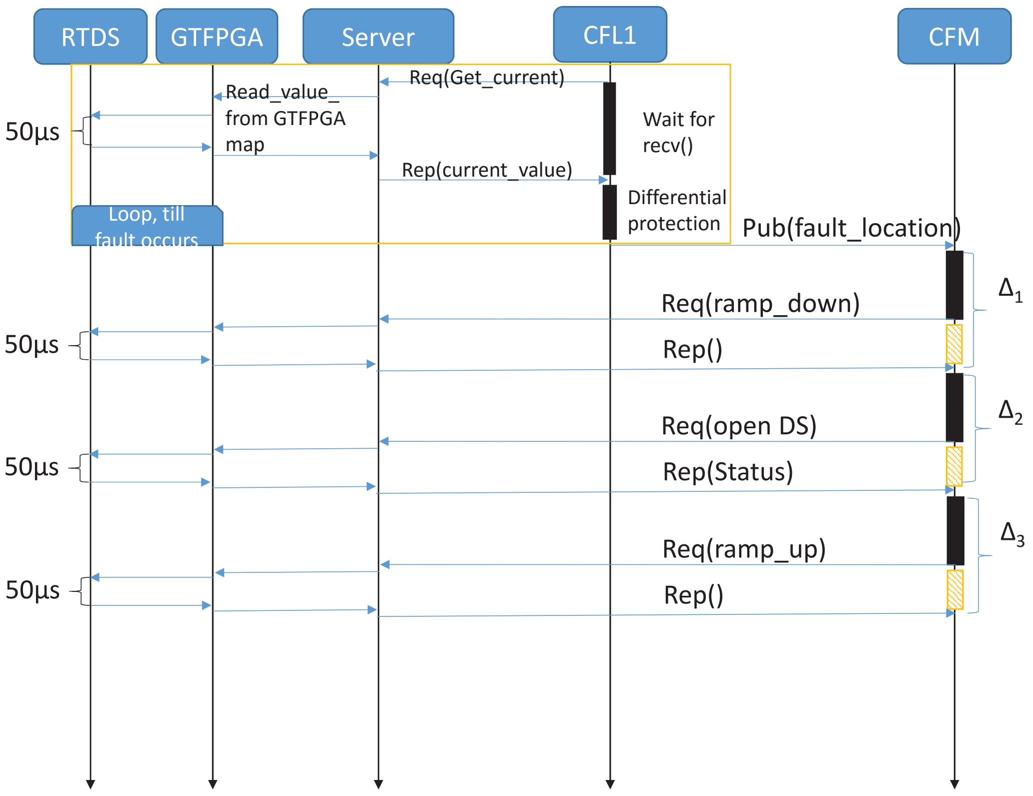


Figure 7.11.: Sequence diagram for CFM approach

During normal operation the CFL monitors the sections. After a faults occurs

Algorithm 7.1 min_dis_isolation_devices(G, visiting, found, visited)

```
Input: Graph G: contains information of the electrical grid + fault_vertex
Input: visiting: contains the the fault_vertex at the beginning
Input: found: used for recursiv function: Init value is False
Input: visited: Init value is a empty list
Output: contains all necessary devices for isolation
 1: involved_isolation_components =
 2: visited = visited + visiting
 3: if (visiting can isolate with current flowing) then
 4:     involved_isolation_components += visiting
 5: else if (visiting can conduct current) then
 6:     if (visiting can isolate with no current flowing) then
 7:         found = TRUE
 8:         involved_isolation_components += visiting
 9:     end if
10:     foreach unvisited n of neighbors(visiting) do
11:         rec_deviceList = min_dis_isolation_devices(G, n, found, visited)
12:         involved_isolation_components += rec_deviceList
13:     end foreach
14: end if
15: return involved_isolation_components
```

it publish the information to the CFM,which immediately starts to perform the recovery sequence. After adding the fault vertex to the graph it starts to compute the involved components and de-energize the system. After the system is de-energized, the CFM sends the command to open the switches. It sends the command until it receives the feedback, that all DS are open. Afterwards the command to re-energize the system is sent.

7.6.4. Distributed Computation

For the distributed architecture the described hardware was used as described in Section 7.3. Also the electrical network was parted in two regions R, as shown in Figure 7.12 and two CFLs were monitoring them.

For each region a CFL system is used to monitor the electrical system. After a fault occurs in one of the CFL regions, the CFL publishes the information to all DFM systems. The DFMs know the entire graph, compute and perform the recovery sequence.

In the corresponding time sequence diagram (Figure 7.13) it is shown that two

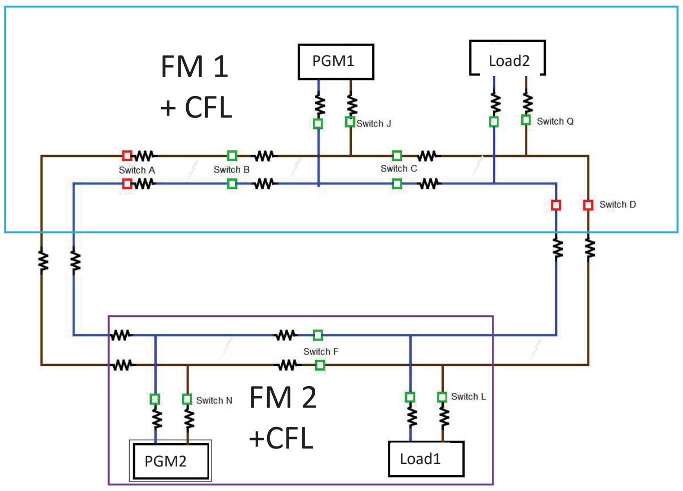


Figure 7.12.: A possible example to split the electrical grid in regions

CFLs are communicating with the server to get the current value and calculate the PDP. If a fault occurs in the monitored region of CFL 2 for example, CFL2 publishes the fault location with a unique ID. FM1 and FM2 starts with the computation of the involved devices, which are the time Δ_1 and Δ_2 and FM1 send the commands to ramp down the PGMs. In the same time FM2 sends the commands to open the particular DS. After the commands are sent, both FM start to calculate the best way to re-configure the system and the first one will send the ramp up commandos, after the fault is isolated. Because both FM are calculating the re-configuration, both systems know the new online topology. The time Δ_3 represents the computation time for the re-configuration and sending the messages and Δ_4 indicates the time for the computation of the re-configuration.

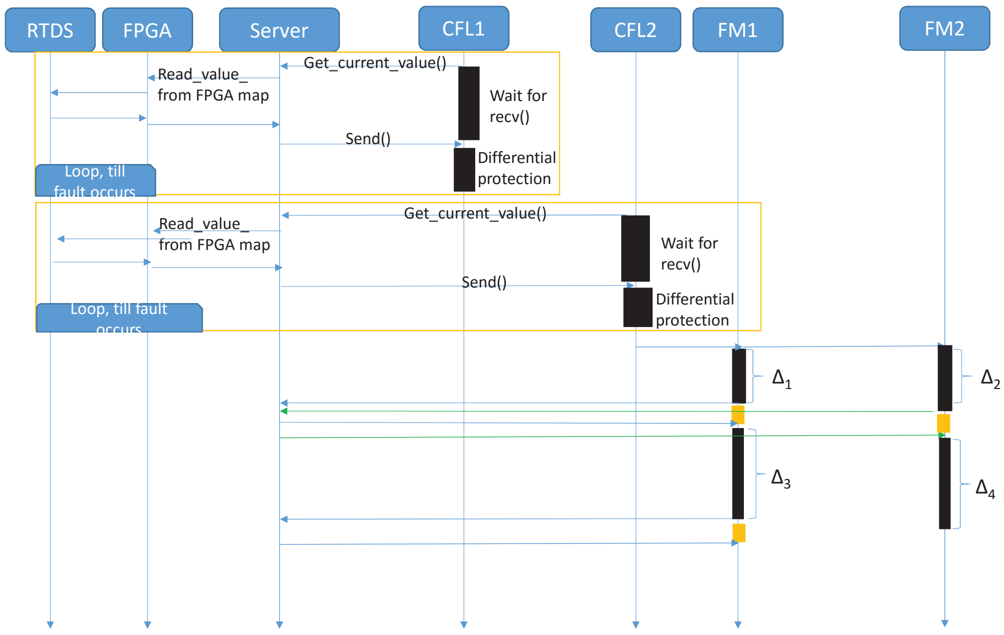


Figure 7.13.: Sequence diagram for DFM approach

8. RESULTS

This chapter presents the results of the performacnes and measurements of the proposed architectures. The goal was to recovery the proposed electrical grid in less then 8 ms. At first the computation of the offline rules are shown. After the sections S are computed a fault is simulated with RTDS in the electrical grid. The FM controller performs the recovery sequence and isolate the fault.

8.1. Computation of the offline rule

With the given connectivity matrix M_c the graph G was created as it is shown in Figure 7.9. Based on the Definition 2 all section can be computed, therefore all subgraphs are calculated, which contain a set of vertices to isolate the fault. The computed subgraphs are:

$$s_1 = [DA, DB]$$

$$s_2 = [DB, DC, DJ, B1]$$

$$s_3 = [DC, DD, DQ, B2]$$

$$s_4 = [DD, DF, DL, B3]$$

$$s_5 = [DF, DN, DA, B4]$$

Accordingly to Algorithm 4.3, a section is taken and the percentage differential protection scheme is computed. If a fault occurs the fault vertex is sent to the FM. Figure 8.1 shows the voltage and current waveforms of the section $s_1$after a fault is simulated in this section with RTDS.

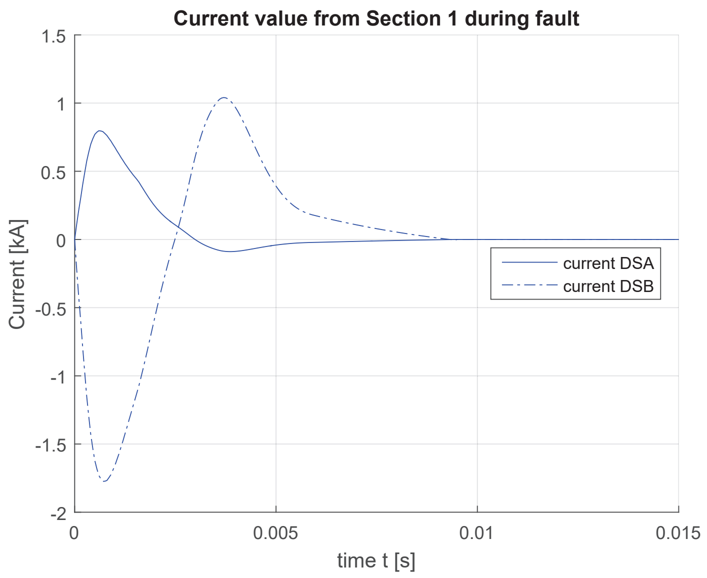


Figure 8.1.: Current in section s_1 after fault is initialized

8.2. Computation of the Recovery Sequence

After a fault has been identified with the CFL program publishes the fault vertex to the FM. The FM adds the fault vertex to the graph. For example, a fault in $s_1$occurs, a fault vertex will be added between DA and DB. This case is illustrated in Figure 8.2.

Now the FM computes the involved components to isolate the fault. Based on Algorithm 7.1 the minimum isolation set is computed. The first time the algorithm is computed, the input values are the graph G, which contains the fault vertex and the $fault\_vertex$. Based on the online configuration the computed isolation set contains following devices:

- PGM1
- PGM2
- DA and DB

With the isolation set the Algorithm 5.1 and 5.2 can be performed to isolate the faults. Because of physical limitation the computation and communication can only take a certain mount of time. This is described in the next section.

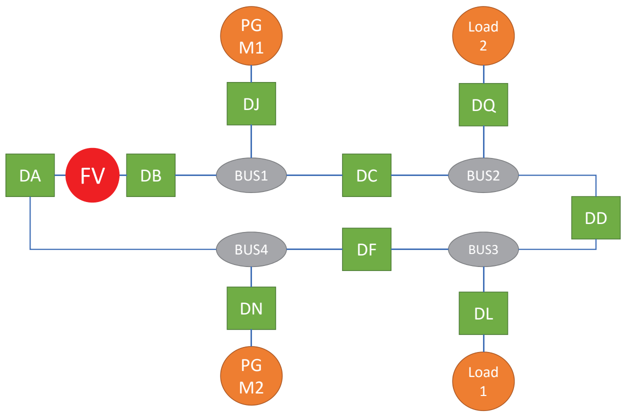


Figure 8.2.: Fault vertex (FV) is added in Section s_1 between disconnect switch DSA and DSB

8.3. Physical constraints

Figure 8.3 illustrates the timeline with the constraints imposed by the electrical devices during a fault recovery sequence.

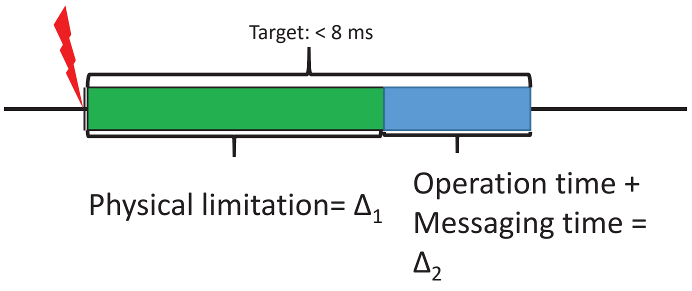

Target: < 8 ms
Physical limitation= Δ_1
Operation time + Messaging time = Δ_2

Figure 8.3.: Timebudget for recovery sequence

The physical limitation can be calculated with the assumption that every device receive the message at the same time as:

$$\Delta_1 \quad = \quad ramp\_down() + open() + ramp\_up() \tag{8.1}$$

- Δ_1 : physical limitation of used devices

- $cmd\_ramp\_down()$: Message from FM to PGM to ramp down $\rightarrow De - energize$ De-energize
- $cmd\_open()$: Message from FM to each DS to open for isolation $\rightarrow Isolation$
- $cmd\_ramp\_up()$: Message from FM to PGM to ramp up $\rightarrow Re - energize$

With Equation 8.1, the computation time and communication time can be calculated as:

$$\Delta_2 \quad = \quad 8ms - \Delta_1 \tag{8.2}$$

- Δ_2 : Time for computation of involved components and sending messages
- $8ms$: goal to isolate fault
- Δ_1 : from Equation 8.1

According to Winkelnkemper et.al papers [7, 46] an MMC is able to ramp down and up in 2 ms and to Sloderbecks paper [15] a ultra fast DS is able to open and close in 1 ms. Therefore, the $physical\_limitation$ accordingly to Equation 8.1 is:

$$\Delta_1 \quad = \quad 2ms + 1ms + 2ms = 5ms \tag{8.3}$$

With the physical limitation the time for the FM can be calculated as:

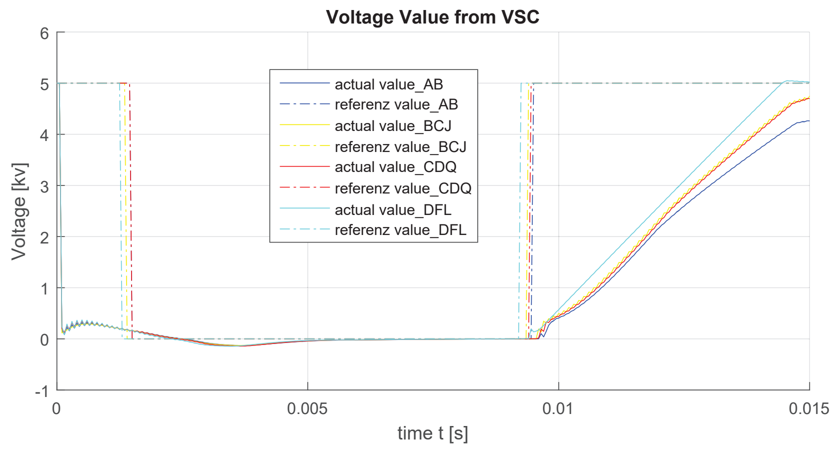
$$\Delta_2 \quad = \quad 8ms - \Delta_1 = 3ms \tag{8.4}$$

In the next sections the recovery sequence based on the CFM architecture and DFM architecture is shown.

8.4. CFM Recovery Results

In this section the results with the CFM approach is shown. As in Equation 8.4 the CFM controller has 3ms time to communicate via the GTFPGA to the simulated electrical devices and compute the involved components for the different fault locations. Depending of the fault location, different isolation sets are computed.

Figure 8.4 shows the voltage and current waveforms of the system during the fault recovery sequence. After the fault is detected and the FM is informed, the FM sends the commands to de-energize as it is shown in Algorithm 5.1. The first chart shows the voltage during a fault recovery for the fault location one, three, four and five. After 1.3 ms the commands to ramp down the voltage source and current source are sent.



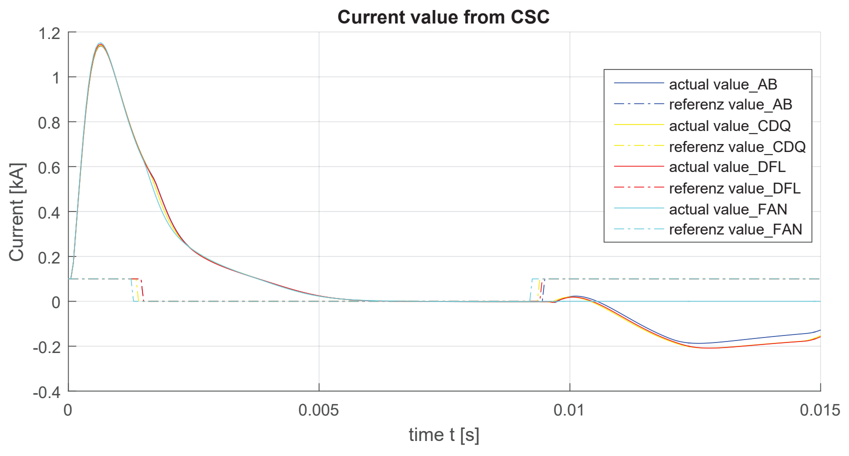


Figure 8.4.: Current and Voltage from CSC and VSC for the fault location 1,3,4 and 5

After the commands for the de-energization part was sent, the commands for opening the disconnect switches are sent. Figure 8.5 shows, that after 1.45 ms the command for opening the particular DS was sent. Until the system is fully de-energized, the DS is not able open. Each DS has its own main control and measures the current and decides to open, if the current value is under a specific value, which is defined by the manufacturer of the DS. The status of DS changes after 9.5 ms.

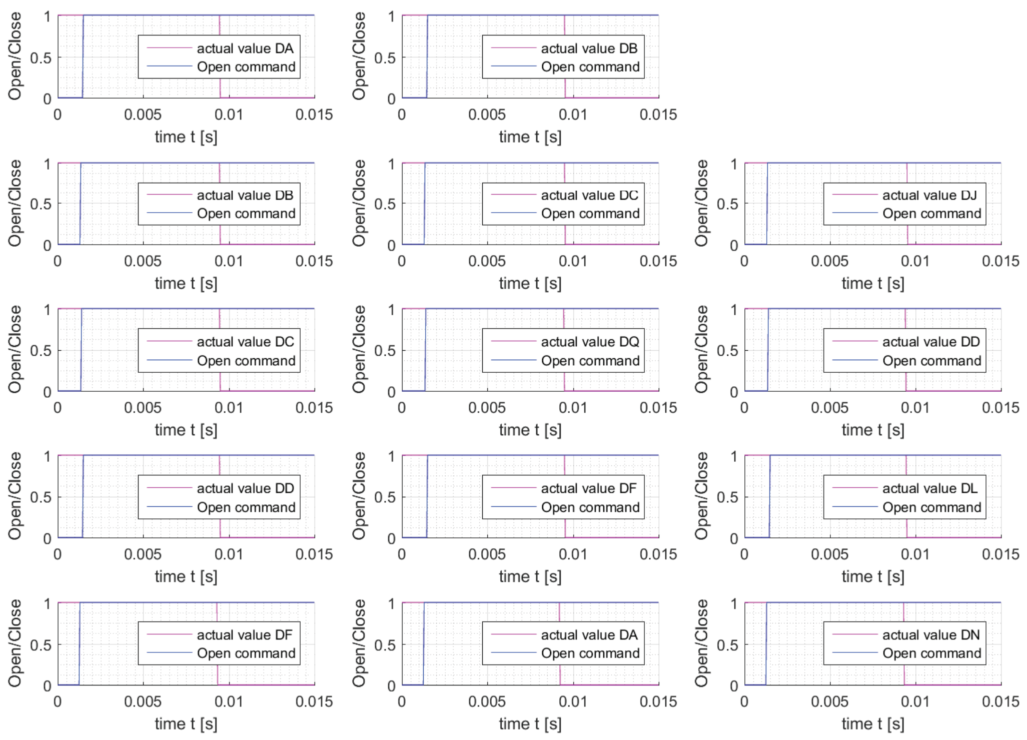


Figure 8.5.: Opening commands for each DS for particular section

Because the re-configuration and re-energize part was not part of this work, a special case happens for the fault section two. After a fault occurs in section 2 the FM isolated the fault with the switches DSB, DSC and DSJ according to Table 7.1. With opening the DSJ the PGM1 was isolated from the system. In Figure 8.6 the FM tried to ramp up the PGM1 and the voltage waveform is showed.

8.5. DFM Recovery Results

In this section the recovery sequence based on the DFM architecture is described. As described in subsection 5.3.2 multiple controller handle the fault recovery sequence. Therefore, the electrical grid was divided into two regions as it is shown in 7.12. A fault was initialized in section $s_4$and isolated with two FM. Because only PGM2 is conducted to the fault, only the current waveform from PGM2 is shown in Figure 8.7. Also the signals for opening the DS is shown.

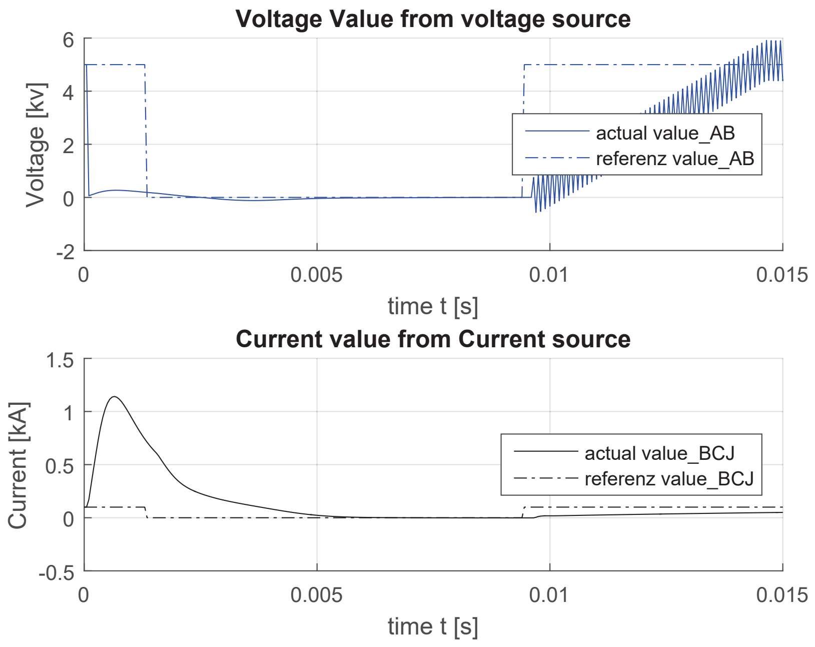


Figure 8.6.: Current and Voltage from CSC and VSC for fault location 2

With the DFM approach it was possible to send the re-energize command after 6.2 ms.

In the next chapter the FM approach is discussed.

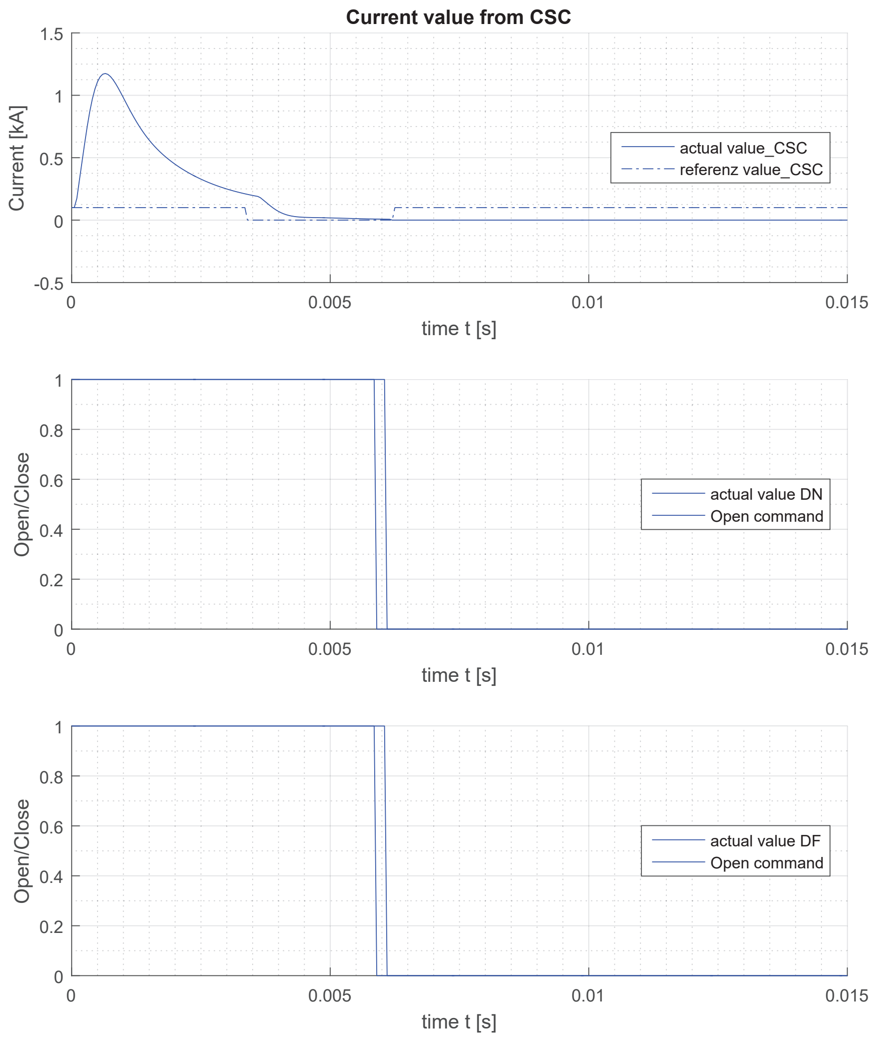


Figure 8.7.: Current waveform from CSC for fault location 4

9. DISCUSSION

In this study, the question under discussion is, if a graph based traversal can automate the detection and isolation of a fault in a Medium Voltage DC (MVDC) Shipboard Power System.

Based on the Xing's dissertation the steps for a fault recovery is shown. At first a fault has to be identified and located in a MVDC system. In Tamaskars thesis an ultra-fast fault location identification approach using a real-time equipment is discussed. These two researches are dealing with the fault identification, but none of them investigate the next step of an automated fault recovery. At first an applicable representation for a MVDC system was necessary to automate the fault identification and recovery. Therefore, I decide to represent the MVDC SPS as a graph to pull from the knowledge of the graph theory. Using the graph approach, rules were defined to transform the electrical network into a graph. The challenge is to transform the electrical network without losing the information of the current flow and the electrical attributes as it is shown in Algorithm 4.2. Therefore, I tried to abstract the electrical devices in a way to be able to transform every device from a MVDC SPS.
After the transformation it was possible to generate rules for the detection of faults and for the isolation.
The generation of the fault detection rules guarantees that each section will be monitored. In previous works, the code for the sections S were hardcoded and for each change in the electrical network the CFL also changed. With the graph approach only the connectivity matrix is used to generate the CFL rule. This provides a less error prone system for customers and facillites during the development of new shipboard power system. It is also better scalable than the previous solution based on the number on lines of codes. Accordingly, to Algorithm 4.3 the number of lines is not increasing even if the number of sections will increase. In summary with the graph representation it is possible to generate the code for the CFL, which is better scaleable than in previous works. It also reduces the

human inputs and prevents for coding faults. Also it facilities, that if the SPS model change, only the update connectivity matrix is needed for fault identification and location. The next step was to develop the automated fault isolation algorithm. The challenge in this step is to find the isolation set and all sources, which supply power to the fault. As the result shows with the DFM architecture the recovery sequence can be done in less than 8 ms. The challenge to develop an communication architecture was, that nothing was defined by previous works. Therefore, the definitions were made to create the interface between the CFL and FM.

10. FURTHER WORK

The experience and results achieved through the approach of recovery a SPS in less then 8 ms suggests future research including: testing approach on real electrical devices 10.1, re-configuration and re-energization 10.2, integrating multiple FMs 10.3, Human Machine Interface (HMI) 10.4, testing the distributed architecture 10.5, analytical model of distributed architecture 10.6 and increasing the efficiency of the developed algorithms 10.7.

10.1. Real Electrical Devices

In this work the shipboard power system was simulated with the Real-Time Digital Simulator. All the devices are based on a average model of the real world device. All the commands for ramping down and ramping up were sent to the RTDS. Therefore, the interface between the electrical devices of a SPS and the FM has to be evaluated.

10.2. Re-configuration and Re-energization

After a fault is isolated the system should be re-energized to operate as *well* as possible. Therefore, the re-configuration should plan how the disconnect switch should be closed or opened to use as much power as possible. For doing this, the P&E system contains necessary information, which system in the SPS are important to get re-energized and which system maybe can be switched off. For example on the SPS, the drive of the ship is necessary and must be re-energized, but the entertainment system on the ship can be switched off. The re-configuration needs therefore, more information about the electrical network. During the graph transformation, these information should be added to the vertices and therefore, the electrical network have to be analysed.

10.3. Integrating Multiple FMs

The FM was tested for a centralized version and a distributed architecture. For the distributed architecture only 2 controls were used. A lager electrical network which can be divided into more regions and more controls of the FM can provide redundancy for the recovery functionality and also increase the performance.

10.4. Human Machine Interface

After a fault is isolated in the system, there is no current flowing to this section. If it is possible to repair the faulted section it should be possible to add this section again to the electrical network. This can be done with a HMI. If adding the recovered section to the electrical grid again, the system should re-configure the system again, based on the re-configuration work from section 10.2.

10.5. Testing the Distributed Architecture

In this work it was shown that the proposed distributed architecture works and increase the performance. The distributed architecture was tested with the same hardware, but has it the ability to interact with different versions of hardware. Thus, it is necessary to test of resilience to failures of network.
Another thing is the saftey of sending messages between the electrical devices, the CFL and FM which should be add that the FM can not be hacked by someone.

10.6. Analytical Model of Distributed Architecture

A distributed architecture was developed, which is able to share computational power and perform the recovery sequence parallel. An analytical model for the reasoning of the redundancy of the approached distributed system should help to find the influence off losing one control to finish the performance of the recovery
.

10.7. Efficiency

The Depth-First Search graph traversal needs a lot of computational resources based on [21]. Pre- computing all possible recovery sequences based on the online configuration of the shipboard power system and saving them into a look-up table would decrease the computational rescources during a fault sequence. After a fault occurs, the program only have to find the entry in the table and perform the recovery sequence. This would decrease the computational power enormously compared to the graph traversal, because to find an entry in a list needs the computational rescources of $\mathcal{O}(1)$. After the fault is isolated, it would be necessary to pre-compute the new table again based on the graph traversal algorithms.

11. CONCLUSION

This work addressed automated recovery process of a MVDC shipboard power system. As a set objective, MVDC shipboard power systems are designed that fault sections can be isolated and the system can continuing work with normal or limited performance again. The fault sections were isolated in less then 8 ms. The recovery was performed automatically. To perform the recovery automatically the electrical network needed to be stored in a data format to use it to computer program. The chosen solution for this problem was the transformation of the electrical network into a graph. The benefit of the graph representation is to pull from the wealth of knowledge and techniques. With the graph representation it was possible to decrease rapidly the number of inputs from humans for fault identification and fault recovery. Only the connectivity matrix was needed for reaching the goal of automated recovery. Therefore, a fault has to be identified in the system. In the literature [4, 5] a centralized fault location and identification (CFL) system was developed. With the graph approach the code for the CFL was generated automatically and the input by human was decreased to one.
After a fault is occurred the FM performed automatically the recovery sequence. With the proposed distributed architecture it was possible to share the computational power and have a redundant system. Based on the graph approach and the assumption, that each controller knew the entire graph, the tasks for the recovery sequence could be done in parallel.
Summing up, a fault management was developed which is able to isolate a fault in less than 8 ms. With the transformation of the electrical network to a graph, it was possible to use the knowledge of the graph theory and use them for the recovery. Finally algorithms were developed, which are scalable and less error prone that in previous works.

Bibliography

[1] . R. Technologies, "Rtds technologies inc. | real time digital power system simulation," https://www.rtds.com/, (Accessed on 08/15/2016).

[2] H. Mirzaee, S. Bhattacharya, and S. Bala, "A high power medium-voltage dc amplifier system," in *2011 IEEE Energy Conversion Congress and Exposition*, Sept 2011, pp. 4043–4050.

[3] A. Ghaderi, H. A. Mohammadpour, and H. Ginn, "Active fault location in distribution network using time-frequency reflectometry," in *Power and Energy Conference at Illinois (PECI), 2015 IEEE*, Feb 2015, pp. 1–7.

[4] X. Liu, "A centralized fault management approach for the protection of smart grids," Ph.D. dissertation, THE FLORIDA STATE UNIVERSITY, 2015.

[5] S. Tamaskar, "Performance analysis of a network based, ultrafast, centralized fault location and identification system," 2016.

[6] A. Lesnicar and R. Marquardt, "An innovative modular multilevel converter topology suitable for a wide power range," in *Power Tech Conference Proceedings, 2003 IEEE Bologna*, vol. 3, June 2003, pp. 6 pp. Vol.3–.

[7] M. Steurer, F. Bogdan, M. Bosworth, O. Faruque, J. Hauer, K. Schoder, M. Sloderbeck, D. Soto, K. Sun, M. Winkelnkemper, L. Schwager, and P. Blaszczyk, "Multifunctional megawatt scale medium voltage dc test bed based on modular multilevel converter (mmc) technology," in *2015 International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles (ESARS)*, March 2015, pp. 1–6.

[8] M. Andrus, H. Ravindra, J. Hauer, M. Steurer, M. Bosworth, and R. Soman, "Phil implementation of a mvdc fault management test bed for ship power systems based on megawatt-scale modular multilevel converters," in *Electric Ship Technologies Symposium (ESTS), 2015 IEEE*, June 2015, pp. 337–342.

[9] J. Langston, M. Andrus, M. Steurer, D. Alexander, J. Buck, G. Robinson, and D. Wieczenski, “System studies for a bi-directional advanced hybrid drive system (AHDS) for application on a future surface combatant,” in *Electric Ship Technologies Symposium (ESTS), 2013 IEEE.* IEEE, 2013, pp. 509–513.

[10] R. Chan, S. Sudhoff, and E. Zivi, “An approach to optimally allocate energy storage in naval electric ships,” in *Electric Ship Technologies Symposium (ESTS), 2011 IEEE*, April 2011, pp. 402–405.

[11] R. Chan, S. Sudhoff, Y. Lee, and E. Zivi, “A linear programming approach to shipboard electrical system modeling,” in *Electric Ship Technologies Symposium, 2009. ESTS 2009. IEEE*, April 2009, pp. 261–269.

[12] A. Cramer, H. Chen, and E. Zivi, “Shipboard electrical system modeling for early-stage design space exploration,” in *Electric Ship Technologies Symposium (ESTS), 2013 IEEE*, April 2013, pp. 128–134.

[13] N. Doerry and J. Amy Jr, “Mvdc shipboard power system considerations for electromagnetic railguns.”

[14] Y. Khersonsky, “New ieee power electronics standards for ships,” in *Electric Machines Tech. Symp.(EMTS)*, 2012.

[15] M. Sloderbeck, H. R. Dionne Soto, M. Steurer, and A. Challita, “Megawatt scale demonstration of high speed fault clearing and power restoration for mvdc systems utilizing fast disconnect switch.”

[16] B. Diaz, T. H. Ortmeyer, B. Pilvelait, M. Izenson, W. Chen, and N. Spivey, “System study of fault current limiter for shipboard power system,” in *2009 IEEE Electric Ship Technologies Symposium*, 2009.

[17] “Ieee guide for protecting power transformers,” *IEEE Std C37.91-2008 (Revision of IEEE Std C37.91-2000)*, pp. 1–139, May 2008.

[18] S. Miao, P. Liu, and X. Lin, “An adaptive operating characteristic to improve the operation stability of percentage differential protection,” *IEEE Transactions on Power Delivery*, vol. 25, no. 3, pp. 1410–1417, July 2010.

[19] M. J. Thompson, “Percentage restrained differential, percentage of what?” in *Protective Relay Engineers, 2011 64th Annual Conference for*, April 2011, pp. 278–289.

[20] N. Biggs, *Algebraic graph theory.* Cambridge university press, 1993.

[21] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*, 2nd ed. McGraw-Hill Higher Education, 2001.

[22] J. Kleinberg and E. Tardos, *Algorithm Design.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[23] L. Lovász and K. Vesztergombi, *Discrete Mathematics: Lecture Notes, Yale University, Spring 1999*, 2000. [Online]. Available: https://books.google.com/books?id=OXObjwEACAAJ

[24] Wikibooks, "A-level computing/aqa/paper 1/fundamentals of algorithms/tree traversal — wikibooks, the free textbook project," 2016, [Online; accessed 1-August-2016]. [Online]. Available: https://en.wikibooks.org/w/index.php?title=A-level_Computing/AQA/Paper_1/Fundamentals_of_algorithms/Tree_traversal&oldid=3092536

[25] S. Even and G. Even, *Graph Algorithms.* Cambridge University Press, 2011. [Online]. Available: https://books.google.com/books?id=m3QTSMYm5rkC

[26] S. Skiena, *The Algorithm Design Manual: Text*, ser. Computer Science: Algorithm Design. TELOS–the Electronic Library of Science, 1998. [Online]. Available: https://books.google.com/books?id=TrXd-gxPhVYC

[27] C. Y. Lee, "An algorithm for path connections and its applications," *IRE Transactions on Electronic Computers*, vol. EC-10, no. 3, pp. 346–365, Sept 1961.

[28] C. E. Leiserson and T. B. Schardl, "A work-efficient parallel breadth-first search algorithm (or how to cope with the nondeterminism of reducers)," in *Proceedings of the Twenty-second Annual ACM Symposium on Parallelism in Algorithms and Architectures*, ser. SPAA '10. New York, NY, USA: ACM, 2010, pp. 303–314. [Online]. Available: http://doi.acm.org/10.1145/1810479.1810534

[29] S. Abdullahi, "An application of graph theory to the electrical circuit using matrix method," *IOSR Journal of Mathematics (IOSR-JM)*, vol. Volume 10, no. Issue 2 Ver. II, apr 2014.

[30] K. Birman and T. Joseph, "Exploiting virtual synchrony in distributed systems," in *Proceedings of the Eleventh ACM Symposium on Operating Systems Principles*, ser. SOSP '87. New York, NY, USA: ACM, 1987, pp. 123–138. [Online]. Available: http://doi.acm.org/10.1145/41457.37515

[31] D. E. Perry and A. L. Wolf, "Foundations for the study of software architecture," *ACM SIGSOFT Software Engineering Notes*, vol. 17, no. 4, pp. 40–52, 1992.

[32] G. Hohpe and B. Woolf, *Enterprise integration patterns: Designing, building, and deploying messaging solutions.* Addison-Wesley Professional, 2004.

[33] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*, 5th ed. USA: Addison-Wesley Publishing Company, 2011.

[34] M. Glinka, "Prototype of multiphase modular-multilevel-converter with 2 mw power rating and 17-level-output-voltage," in *Power Electronics Specialists Conference, 2004. PESC 04. 2004 IEEE 35th Annual*, vol. 4, 2004, pp. 2572–2576 Vol.4.

[35] M. Glinka and R. Marquardt, "A new ac/ac multilevel converter family," *IEEE Transactions on Industrial Electronics*, vol. 52, no. 3, pp. 662–669, 2005.

[36] M. Hagiwara and H. Akagi, "Control and experiment of pulsewidth-modulated modular multilevel converters," *IEEE Transactions on Power Electronics*, vol. 24, no. 7, pp. 1737–1746, July 2009.

[37] M. Hagiwara, K. Nishimura, and H. Akagi, "A medium-voltage motor drive with a modular multilevel pwm inverter," *IEEE Transactions on Power Electronics*, vol. 25, no. 7, pp. 1786–1799, July 2010.

[38] J. Kolb, F. Kammerer, M. Gommeringer, and M. Braun, "Cascaded control system of the modular multilevel converter for feeding variable-speed drives," *IEEE Transactions on Power Electronics*, vol. 30, no. 1, pp. 349–357, Jan 2015.

[39] X. Shang, G. Wang, F. Li, Q. Wu, and J. Feng, "Low output frequency operation of modular multilevel matrix converter," in *2015 5th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies (DRPT)*, Nov 2015, pp. 2259–2264.

[40] M. Winkelnkemper, A. Korn, and P. Steimer, "A modular direct converter for transformerless rail interties," in *2010 IEEE International Symposium on Industrial Electronics*, July 2010, pp. 562–567.

[41] D. Jansen and H. Buttner, "Real-time ethernet the ethercat solution," *Computing Control Engineering Journal*, vol. 15, no. 1, pp. 16–21, Feb 2004.

[42] F. S. Univeristy, “Tour the center for advanced power systems,” http://www.caps.fsu.edu/tour.html, August 2016, (Accessed on 08/10/2016).

[43] P. S. Foundation, “Socketserver - a framework for network servers - python 3.5.2 documentation,” https://docs.python.org/3.5/library/socketserver.html, July 2016, (Accessed on 08/12/2016).

[44] V. Corp, “Versalogic - mamba (vl-ebx-37) - ebx sbc with intel core 2 duo,” http://www.versalogic.com/mam, 2016, (Accessed on 08/10/2016).

[45] iMatix Corporation, “Distributed messaging - zeromq,” http://zeromq.org/, 2014, (Accessed on 08/15/2016).

[46] P. Blaszczyk, M. Winkelnkemper, and L. Schwager, “Converter energy balancing in mmc system energy sharing using master controller,” in *Electrical Drives and Power Electronics (EDPE), 2015 International Conference on*, Sept 2015, pp. 30–37.

List of abbreviations

MMC	Modular Multilevel Converter
FM	Fault Management
SPS	Shipboard Power System
CFL	Centralized Fault Location and Identification system
ι	Isolation Set
Req/Rep	Request/ Reply
Pub/Sub	Publisher Subscriber
CFM	Centralized Fault Management
DFM	Distributed Fault Management
PGM	Power Generation Module
DS	Disconnect Switch
B	Bus

List of Figures

List of Tables

List of Algorithms