Dual Path Communications over Multiple Spanning Trees for Networked Control Systems

Sylvain Kubler[a,*], Jérémy Robert[a], Jean-Philippe Georges[a], Éric Rondeau[a]

[a]*Research Centre for Automatic Control of Nancy, Campus sciences, BP-70239, F-54506 Vandœuvre-lès-Nancy Cedex, France*

Abstract

The switched Ethernet networks are more and more deployed in industry. The Spanning Tree Protocol implemented in the switches enables management of the link connectivity. But the reconfiguration time of the Spanning Tree Protocol (STP) when link failure occurs is not adapted to satisfy industrial constraints. The objective of this paper is to propose a method based only on standard, mitigating the probability of disconnection between nodes having hard real-time properties. The approach developed in this paper consists of duplicating frames and of forwarding them on different paths. These paths are optimized and specified by using genetic algorithms. OPNET simulations show the interest of this proposal on a particular Networked Control System.

Keywords: Real-time systems, Fault tolerance, Spanning-tree, Genetic Algorithm, Switched Networks

1. Introduction

The modern plant architectures implement communication networks to control and to monitor their remote and distributed applications. The major interest facing the point to point architectures is to mitigate the wire costs and to enable easier information sharing. In the framework of embedded systems and industrial systems, the communication requirements are to guarantee bounded end-to-end delays and to maintain the connectivity between all network nodes. In the 1980's, many networks named fieldbuses were developed to respect these strong constraints. More recently, the trend in fieldbuses was to use Ethernet protocol (IEC 61784 gathers the different fieldbuses standards). The advantages are that Ethernet is a well known protocol, widely implemented (ensuring its timelessness), and its performance is continually increasing with technology evolution (especially its bandwidth). Consequently, Ethernet is not a determinist network since it implements the carrier-sense multiple access with collision detection (CSMA/CD) mechanism to resolve the problem of contention in case of simultaneous data transmission [1]. However, the fully switched architectures allow inhibiting the collisions and making possible delay estimations in the worst cases [2]. Moreover, the IEEE 802.1d introduces algorithms to face loops' issues when redundant links are used. In such cases, the Spanning Tree Protocol (STP) defines a hierarchical logical tree topology by inhibiting some switch ports. STP is then able to react to a link failure by activating inhibited ports. Nevertheless, the standard STP reconfiguration time period (in seconds) is too long and is not compatible with the industry requirements (often less than $1s$). This is especially the case for Networked Control System where a network is used in feedback control loops. In this field, several research works [3] focus on the way to reconfigure the network when failures occur, by proposing either new protocols than STP (such that native devices are not anymore supported) or specific/dedicated physical network organizations. The goal of this paper is both to retain only native protocols (like STP) and to significantly increase the network dependability by reducing the reconfiguration time, even to zero in particular cases. This dependability is evaluated in terms of probability of disconnection between nodes. The practical approach is to duplicate the messages with real-time properties and to send them to the remote nodes by several paths. If one path fails, messages will still arrive at the destination by another path in hiding the STP reconfiguration time period on the trouble path.

This article is organized in the following way: Section 3 describes the Spanning Tree Protocol standards and the related works. Section 4 formalizes the fitness value to be optimized. Section 5 presents the approach based on genetic algorithms to increase the dependability of switched network architectures. Finally, Section 6 illustrates the interest of this study in the Networked Control Systems framework.

2. Networked Control Systems

Feedback control systems in which the control loops are closed through a real-time network are called Networked

*Corresponding author

Email addresses: Sylvain.Kubler@cran.uhp-nancy.fr (Sylvain Kubler), Jeremy.Robert@cran.uhp-nancy.fr (Jérémy Robert), jean-philippe.georges@cran.uhp-nancy.fr (Jean-Philippe Georges), Eric.Rondeau@cran.uhp-nancy.fr (Éric Rondeau)

Control Systems (NCSs) [4]. Many examples of NCSs can be found in several areas: teleoperation systems [5, 6, 7], automotive industry [8] and in automated manufacturing systems [9]. [10] gives an overview and addresses some key issues of the NCS framework, such as limitations in terms of packet-rate, sampling, network delay, and packet dropout. Profibus, CAN, Ethernet, EtherCat, Powerlink are candidate networks for NCS implementations. The characteristics of time delays can be constant, bounded or random according to the network protocol adopted. Consequently, the overall performance of the networked-based control system can be significantly affected by network delays. Figure 1 depicts the NCS synoptic.

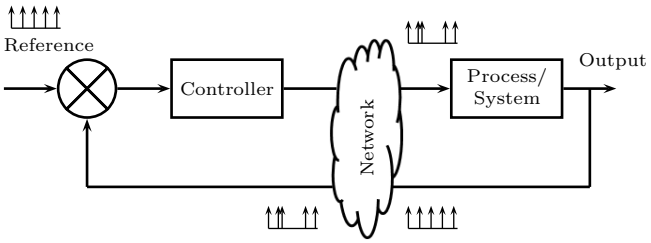


Figure 1: NCS feedback control loop

The main challenge for NCS, where communications between controller, sensor and actuator are performed through a network, is to ensure the quality of service. Indeed, the performance quality of the whole network system is very sensitive to delays, jitters and packet losses. As previously stated, this paper focuses on implementing Ethernet protocol into NCSs. Real-time control via Ethernet is a practical and feasible solution to NCS design [11, 12, 13] and has been the subject of many researches lately [14, 15, 16]. Although several results have been proposed to manage the delays induced by the network, few papers have studied the availability issue to the best of our knowledge. In a network, the robustness of the architecture is increased by adding redundancy. In Ethernet, it corresponds to switched networks as presented in following sections.

3. Spanning Tree

3.1. Introduction

Basically, the dependability of network wires is achieved by implementing redundant links and/or network devices. In Ethernet, the redundancy may generate infinite loops in the physical architecture which can lead to infinite retransmissions needlessly consuming bandwidth. Figure 2(a) illustrates this problem and describes the behavior of a broadcast frame sent by the switch S1 to the switch S4.

This frame is sent to all the output ports of S1. S2 and S3 receive it and forward it to all their output ports. S2 and S3 receive the frame again in another port and forward it to all their output ports, and so on. Thus, the switch S1 twice receives the same frame that it had initially sent. The Spanning Tree Protocol implemented in the switches, aims at breaking the loops in order to form a tree by disabling some ports. STP has also to maintain the network connectivity. Then, STP continually monitors the physical links, and dynamically adjusts their states (enable, disable) according to the link failure occurrences. In the Figure 2(b), the link between S2 and S3 is disabled by STP, and the broadcast frame sent by S1, is forwarded only one time to all the enabled links. If the link S2-S1 fails, STP automatically reactivates the link S2-S3. One of the tree architecture drawbacks is that some redundant links are not used to transmit application frames and that communications along the tree are not necessarily following the shortest path.

It is also important to note that the basic spanning tree is defined independently of traffic origins and destinations. Figure 3 shows then how such optimization may be achieved. Indeed, in the Figure 3(a) S4 and S5 send frames to S1 and S3, respectively. The link S1-S2 supports all the communications, and the disabled link S2-S3 does not transmit any messages. One way to balance the traffic on two links on switched Ethernet is to implement either MSTP (Multiple STP, IEEE 802.1s) or PVST (Per-Vlan STP, Cisco proprietary protocol). The aim of these protocols is to offer the possibility of defining as many trees as possible as Virtual Networks (VLAN, IEEE 802.1q). In Figure 3(b), S1 and S4 are defined in the VLAN 1, S3 and S5 are in the VLAN 2. The MSTP is activated and disables the link S2-S3 from the tree used by the VLAN 1. In the same way, MSTP disables the link S1-S3 from the tree associated with the VLAN 2. The interest of these two stacked trees (at it is shown Figure 3(b)) is to balance the traffic load between links S1-S2 and S2-S3.

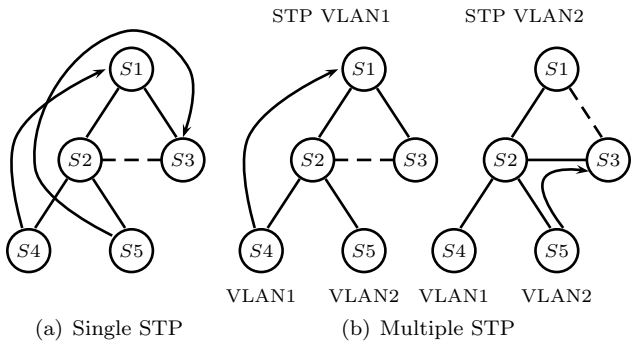
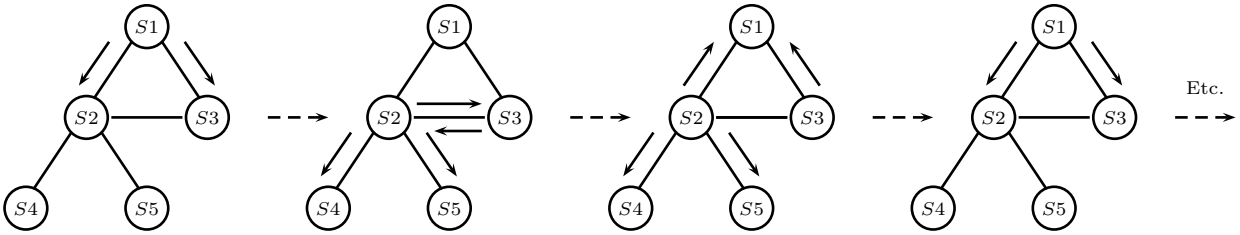


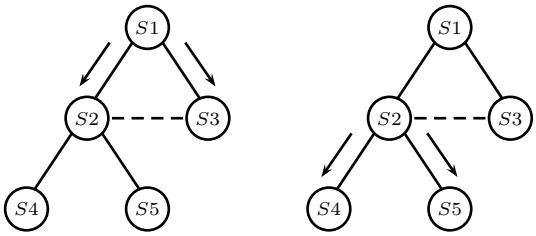
Figure 3: Single STP *vs* Multiple STP

3.2. Reconfiguration time

As described previously, the STP issue is the reconfiguration time period to propose a new fit tree. This time period is between 30 s and 50 s. It corresponds to the addition of three times. Firstly, the waiting time to receive BPDU (Bridge Protocol Data Units), of about 20 s. Secondly, the listening time for each interface subjected to change of state of 15 s. Thirdly, let us note



(a) Loops occurrences without STP



(b) Broadcast traffic behavior with STP

Figure 2: Broadcast traffic behavior

the interface reconfiguration time of about 15 s. The new Rapid STP (RSTP, IEEE 802.1w) mitigates this reconfiguration time and enables the proposed new physical architecture in less than 5 s. However, these times may still not be adapted to real-time environments. The MOXA company has then designed a new mechanism named *Turboring*, which ensures an Ethernet reconfiguration time period around 20 ms. But this solution is not standardized and it is suitable only for a particular topology (a ring). The objective of this paper is to define a method to avoid this discontinuity of service by eliminating it.

3.3. Related works

In [17], an on-line method is defined in packet-switched networks. It consists of a new disjoint multipath routing algorithm (*SimCT*) based on the Colored Graph Tree Theory and takes into account node failures to maintain the communication continuity between a node and a sink. The interest of [17] is to decrease the path length compared, with other proposals and to reduce the number of protocol frame exchanges. Another approach consists of anticipating off-line, the path failure. The PRP solution (IEC 62 439) proposed by ABB [18] consists in using both independent Ethernet networks on which end nodes replicate frames. Duplicated frames are discarded by the receiving nodes. [19] proposes a similar method based on industrial Ethernet (Powerlink) without switches (hub architecture) and consists of duplicating the medium and sending the messages twice, respectively, to both mediums. However, a mechanism, called *link selector* in [19] and *link redundancy entity* (LRE) in the PRP solution is implemented inside each node to ensure incoming information consistency and to duplicate sent messages. Another field of research, similar to [20], deals with 2-layer based auto-configuration of topologically addressed redundant network structures. In [20] an inherent dynamic redundancy management technique is defined for loop prevention. Lee et al. [21] propose a design scheme for cost-effective and reliable Ethernet ring protection networking. However, this solution is dedicated for carrier-grade optical Ethernet networking and relies on a specific network architecture. Qiu et al. [22] develop a solution named Fast Spanning Tree Reconnection (FSTR) to reconnect the spanning tree when a failure occurs, but this solution causes a network service interruption. The same authors define in [23] a local restoration based on multiple spanning trees. Upon failure of a single link, the upstream switch locally restores traffic to preconfigured backup spanning trees. Two approaches are studied, one based on an integer linear programming model and another on heuristic algorithms. Fencl et al. [24] propose a genetic algorithm which aims at designing independent paths between the nodes to satisfy the demands regarding the maximal delay in each communication path, the minimal acquisition costs, while preserving the reliability (e.g. redundant communication path). Finally, the reader could also refer to [25] which provides a survey on a large range of networking applications ranging from the minimal constraint to the rigorous demands of industrial Ethernet networks.

These are related works, but our approach differs from to the network technology. Firstly, we study switched Ethernet architectures not considered in [17, 19]. Secondly, our main objective is to maintain the continuity of service in respect of the standards defined for switched Ethernet, without defining additional STP mechanisms [23]. The work developed by [24] is quiet similar to our proposition, but their fitness function is expressed according to time constrained and not on a reliable criterion. More-

over, our chromosome modeling avoids the generation of unviable solutions in opposition to [24]. In our approach, we rely on the MSTP protocol to define several trees for interconnecting NCS equipment. This strategy consists of implementing a static procedure of forwarding as many frames as defined trees. Thus, if a path is down, the sink node receives the information by at least one of the others paths.

4. Formalization

4.1. Introduction

The first work is to define an expression formalizing the problem of the redundant paths between a source generating traffic and its destination node. A path consists of a series of network components. A network component represents either a network node (switch) or a link. The failure probability of a path depends on the number of network components constituting the path. More network components that are used, higher the path failure probability. Moreover, the failure probability of redundant paths can be correlated when a same network component is used in several paths. Failure of this network component affects the behavior of not only one path but of several paths. Hence, network dependability evaluation depends on the path length (Section 4.2) and on the number of common network components implemented in several paths (Section 4.3). In the present paper, each network component is assumed to have the same failure probability.

4.2. Network dependability with independent paths

We define α_i as being the number network components composing the path i. A network component is a link or a device network (switch). Let λ be the failure probability per hour of any network component and $\mu = 1 - \lambda$ as being the non-failure probability. The failure probability of a path i (P_{Path_i}) is given by (1). The failure probability of the network (P_{Net_j}) composed of j independent paths is given then by (2).

$$P_{Path_i} = 1 - (1-\lambda)^{\alpha_i} = 1 - \mu^{\alpha_i} \quad (1)$$

$$P_{Net_j} = \prod_{i=1}^{j} P_{Path_i} = \prod_{i=1}^{j} (1 - \mu^{\alpha_i}) \quad (2)$$

The assessment of (2) is studied according to the SIL specifications (Safety Integrity Level, IEC 61508 standard). Table 1 provides the values used in the continuous mode. Our research focuses on Networked Control Systems where the SIL 3 and SIL 4 are the levels generally required, especially in avionics and nuclear power plants. The analysis is based on the choice of network components with SIL 4 (with $\lambda = 10^{-8}$). The aim is to observe whether the whole network (constituted of SIL 4 network components) respects the SIL 4 constraint according to both the number of paths and the number of network components used inside a path.

Table 1: Definition of SIL in the continuous mode (IEC 61508)

SIL	Range of λ (Failures by hour)
4	$10^{-9} < \lambda < 10^{-8}$
3	$10^{-8} < \lambda < 10^{-7}$
2	$10^{-7} < \lambda < 10^{-6}$
1	$10^{-6} < \lambda < 10^{-5}$

The results are given in Figure 4. The first observation is the network with one path is quickly degraded and the scores are upper SIL 4. The second is that a network with two independent paths gives a good result since the SIL 4 constraint is guaranteed up to 10 000 network components. Finally, the multiplication of paths considerably increases the cost of the network in terms of maintenance and complexity without necessarily improving the solution relative to SIL constraints. In conclusion, a network architecture implementing two paths is a good arrangement between dependability and cost. Consequently the next section is the study of the analysis redundancy of two paths by considering overlapping effects.

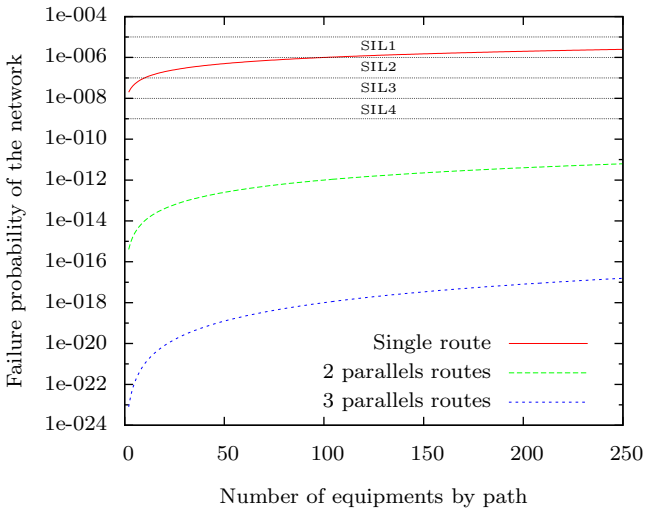


Figure 4: Analysis of dependability for independent paths

4.3. Overlapping paths

In this section, a network consists of two overlapping paths. Let α_i be the number of exclusive network components composing the path i. An exclusive network component belongs only to one path. Let β be the number of common network components used in both paths. As a matter of course, the total length of a path i equals $\alpha_i + \beta$.

Consider the illustrative network shown in Figure 5. Two paths are sharing a common sub path. We note, on

the one hand, that $\alpha_1 = \alpha'_1 + \alpha''_1 = 8$ and $\alpha_2 = \alpha'_2 + \alpha''_2 = 6$ while on the other hand, $\beta = 3$. Path 1 consists of 11 components and the path 2 consists of 9 components.

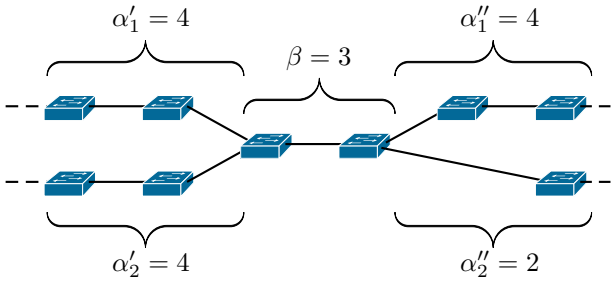


Figure 5: Overlapping of two subpaths

By relying on the equation (2), the whole network failure probability can be determined by (3) which is defined as our fitness value.

$$Fitness = 1 - \left(1 - (1 - \mu^{\alpha_1})(1 - \mu^{\alpha_2})\right)\mu^{\beta} \qquad (3)$$

Let us note that the fitness function parameters α_1, α_2 and α_β are computed thereafter with the help of a function named `Path` (detailed in section 5.2). Figure 6 gives the failure probabilities defined in (3) according to the path length and the rate of the common network components used in the both paths: $\beta/(\alpha_i + \beta)$. Figure 6 shows that when at least one of common network components fails, the whole network is strongly affected. As a result, performances are immediately under the SIL 4 constraint.

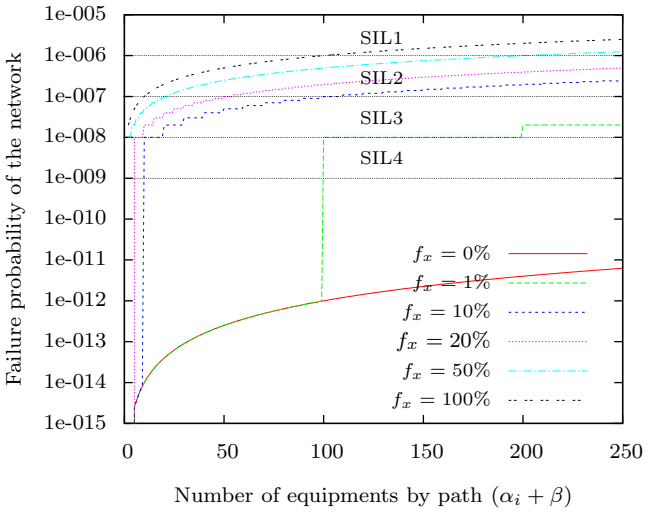


Figure 6: Failure probability for two paths, according to an overlapping percentage $f_x = \frac{\beta}{\alpha_i + \beta}$

Equation (3) will act as the expression formalizing the problem of the redundant paths between a source generating traffic and its destination node. In the following, the issue will be to search for pair of trees such that the paths between two nodes minimize the fitness value. Let L be the number of links and N be the number of nodes in a graph. Firstly, consider a cyclic graph (where $L = N$ for $N \geq 3$) which is a simple case. The search space for one tree is equal to N. As a result, the search space of our problem which consists by exploring all pair of trees among the total number of trees might be limited to C_N^2. Secondly, consider the complex case of a complete graph (where $L = \frac{N(N-1)}{2}$ for $N \geq 3$). This time, the search space for one tree is equal to N^{N-2} as detailed in [26]. As a result, the search space of our problem might be equal to $C^2_{N^{N-2}} = \frac{N^{N-2}\left(N^{N-2}-1\right)}{2}$. That is clearly a combinatory problem. Moreover, this issue is a subclass of problem studied in [17] which is described with complexity order $O(|L||N|)$.

4.4. Conclusion

The expression (3) is a general expression of (2) with two paths. As stated previously, (3) will be used as fitness value in our heuristic (described in the rolling section) to find optimized pair of trees. The network consists of many links, and many nodes such that the search for pair of trees quickly meets an explosion combinatory problem. Thereby, we propose as suggested in [17], to use heuristic algorithms.

5. Optimization algorithms

5.1. Choice of heuristic

There are three main classes of heuristics: the constructive methods (greedy algorithm, pilot method), the local research methods (simulated annealing, taboo search) and the evolutionary methods (Genetic Algorithm, ant colony optimization). It is not easy to compare these methods, but an analysis in [27] between the simulated annealing, the taboo search, the ant colony optimization and the genetic algorithms (GA) provides diagrams giving the probability that one method is better than another. In these works, we notice that for a similar problem, the GA gives the best results. Moreover, GAs are used for network optimization in many research works [28, 29].

5.2. Coding of the component networks

The network architecture is defined by a graph $\mathcal{G} = (\mathcal{S}, \mathcal{M})$ (a digraph or undirected graph) where $\mathcal{S}$ is the set of nodes, i.e. the vertices of the graph and where $\mathcal{M}$ is the incident matrix and which defines the network links. The matrix $\mathcal{M}$ stands for the "incident-vertex-edge matrix" of dimension $m \times n$ where m is the number of vertices and n is the number of edges. This matrix corresponds to the network architecture which remains fixed. Each edge from the graph is numbered such that considering an edge i interconnecting two vertices k and l, we have $\mathcal{M}(j,i) = 1 \ \forall j = k,l$ and $\mathcal{M}(j,i) = 0$ otherwise. As shown in Figure 7, this matrix might be summarized in a vector where each row corresponds to one edge of $\mathcal{G}$ and where the columns give the edges of a given vertex.

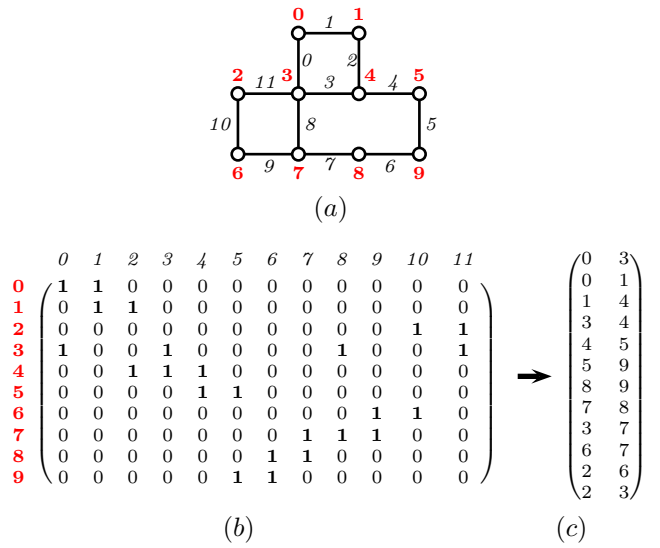


Figure 7: Network modeling, (a) graph $\mathcal{G} = (\mathcal{S}, \mathcal{M})$, ($b$) incident matrix $\mathcal{M}$ and (c) its equivalent vector

Let us focus on the coding of spanning trees. There are several methods of spanning tree representation of graph [30, 31, 32, 33, 34, 35, 36]. [37] explains the coding method based on the representation in the co-tree (i.e. digital coding of open branches designed in [33]). Indeed, to build a tree, it is necessary to reject several links between nodes such that potential loops are eliminated. The interest of coding the open branches is to limit the size of the coding compared to the elementary coding in which each branch is identified with '0' or '1' if it is opened or not. The remaining issue deals with the selection of the branches to open. Since a tree aims here at eliminating potential loops, the solution consists in analyzing each fundamental loops [35]. A fundamental loop is defined such that it does not contain other loops. For each fundamental loop, the principle is then to open one of the edges of the loop. However, it is necessary to check a given branch is not opened in two distinct fundamental loops (since otherwise, it will not guarantee a fitted tree). Each branch is identified by a digit. The "digital coding of opened branches" in each fundamental loop consists finally in a vector composed by the digit of the opened branches. Accordingly, the digital coding of open branches in each fundamental loop is implemented in our algorithm which serves as basis for the coding of the population members. It will avoid the determination of fundamental loops allowing the validity of the resultant topologies after crossover and mutation processes in the genetic algorithms to be tested.

Considering the initial graph shown in Figure 8. First, the fundamental loops in this graph are L_1:*0-1-2-3* (where *0* stands for the branch *0*), L_2:*3-4-5-6-7-8* and L_3:*8-9-10-11*. To note here that the loop *3-4-5-6-7-9-10-11* is not fundamental. Secondly, Figure 8 shows two choices of opened branches. For individual 1, branches *1*, *6* and *8* are respectively opened for loops L_1, L_2 and L_3, whereas for individual 2, the opened branches are *0*, *3* and *10*. Finally, it gives two possible spanning trees (i.e. population members) modeled by the sequence $1, 8, 6$ (Individual 1) and $0, 3, 10$ (Individual 2).

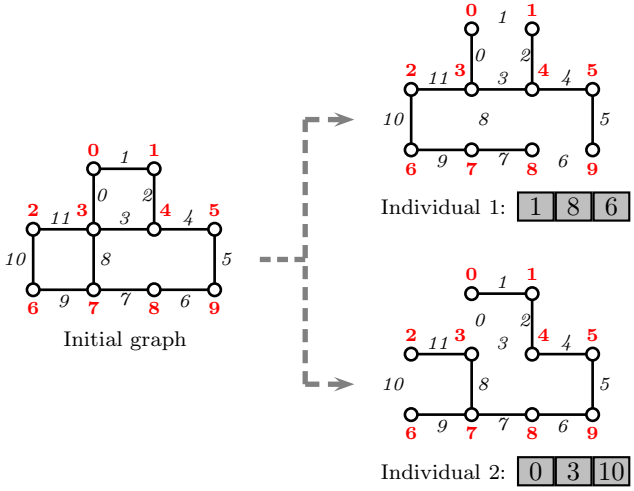


Figure 8: Coding of two population members based on the "digital coding of open branches"

According to this definition, it is possible to propose two algorithms aimed at checking if the graph is connected and identifying the nodes along a path. There are two types of algorithms for going through a graph, the Depth First Search (DFS) and the Breadth First Search (BFS). The difference is related to the analysis sequence of the vertices. The DFS explores "deeply" the paths one by one, whereas the BFS generates a research tree being made layer by layer. However, the BFS requires a lot of memory to store all the alternatives for each layer. Therefore, the DFS algorithm is implemented in our work.

Firstly, an algorithm is defined in order to check that a vertex s from the graph $\mathcal{G}$ is able to access any vertices of $\mathcal{S}$. Running `Accessible_Nodes`$(\mathcal{G}, s, \{\emptyset\})$ from Algorithm 1 returns the list $\mathcal{V}$ of the accessible vertices from s. Hence if $\mathcal{V} = \mathcal{S}$, it means that a graph is connected. This algorithm will then be used to check that the proposed trees are respectful from the spanning tree definition.

Algorithm 1: `Accessible_Nodes`$(\mathcal{G}, i, \mathcal{V})$

output: the list $\mathcal{V}$ of the accessible nodes from i

begin
- $\mathcal{V} \longleftarrow \mathcal{V} \cup \{i\}$;
- **forall the** $j \in \mathcal{S} \backslash \mathcal{V}$ **do**
  - **if** $\exists k \mid \mathcal{M}_{i,k} = \mathcal{M}_{j,k} = 1$ **then**
    - `Accessible_Nodes`$(\mathcal{G}, j, \mathcal{V})$;

Secondly, the function `Path`$(\mathcal{G}, i, e, \mathcal{V}, \mathcal{P})$ of Algorithm 2 is proposed to compute the path length. Indeed, running `Path`$(\mathcal{G}', s, e, \{\emptyset\}, \{\emptyset\})$ returns a list $\mathcal{P}$, containing the vertices of the graph $\mathcal{G}' = (\mathcal{S}, \mathcal{M}')$ along the path between s

and $e \in \mathcal{S}$. Hence, the path length is given by $\mathcal{P}$. This function is useful for determining the fitness function parameters (α_1, α_2 and β) established in the section 4.3, i.e. the number of exclusive and common network components composing both paths. Thus, the `Path` function is a part of the fitness one.

Algorithm 2: `Path`$(\mathcal{G}, i, e, \mathcal{V}, \mathcal{P})$

output: the list $\mathcal{P}$ of the nodes along the path from e to i

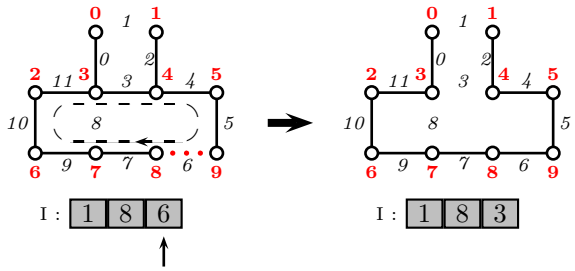
begin
- $\mathcal{V} \longleftarrow \mathcal{V} \cup \{i\}$;
- **if** $i = e$ **then**
  - $\mathcal{P} \longleftarrow \mathcal{P} \cup \{e\}$;
- **else**
  - **forall the** $j \in \mathcal{S} \backslash \mathcal{V}$ **do**
    - **if** $\exists k \mid \mathcal{M}_{i,k} = \mathcal{M}_{j,k} = 1$ **then**
      - `Path`$(\mathcal{G},j,e,\mathcal{V},\mathcal{P})$;
  - **if** $\mathcal{P} \neq \{\emptyset\}$ **then**
    - $\mathcal{P} \longleftarrow \mathcal{P} \cup \{i\}$;
- **return** $\mathcal{P}$;

5.3. Genetic Algorithm

Based on observations expressed in Charles Darwin's "Origin of the Species" Darwin introduces the theory of evolution. Figure 9 gives an overview of the GA used in this paper. It illustrates the natural selection principle where a set of individuals will pass through different stages. Each individual corresponds to a candidate spanning tree solution and is represented by a sequence number of the opened branches, so-called "chromosome". The gray boxes in Figure 9 stand for the genetic algorithm stages. Each stage consists in processing (e.g. evaluating, mating, selecting) a given set of individuals and gives rise to new set of individuals. Information flux between stages are hence set of individuals (i.e. spanning tree solutions).

According to the type of coding previously cited, the crossover and mutation operators might generate more or less invalid individuals compared to the topological constraints. [37] develops a crossover and mutation strategy based on the graph theory approach. The mutation technique consists of randomly picking a gene from a chromosome (representative of an opened branch) and then to determine the formed loop resulting of the closure of this branch. Finally GA randomly selects within this loop, one branch to open. This principle is illustrated Figure 10.

The crossover technique illustrated in Figure 11, consists in randomly picking a reference point in the chromosome of the individual I1, and then determining the loop to be formed after the closure of this branch. GA looks for branches which might be exchanged between the 2 co-trees. Moreover, it is necessary to analyze the chromosome of the individual I2. If there are genes belonging to the



Step 1 : Random choice of one gene (branch) to mute

Step 2 : Random choice of one branch resulting from the formed loop next the closure of the branch 6. Loop: $[3, 4, 5, 6, 7, 9, 10, 11]$

Figure 10: Mutation technique

loop, GA has to make the crossing with one of them. It is necessary to repeat this step until reaching the last gene. Let us notice that the crossover and mutation processes do not give 100% of fitted individuals. However, [37] brings into play the *co-cycle* notion in order to identify quickly and certainly the good branch for swapping, although it does not guarantee that resulting individuals will be in line with the topology constraints. In this paper, the *co-cycle* approach is not implemented and not conformed individuals are excluded.

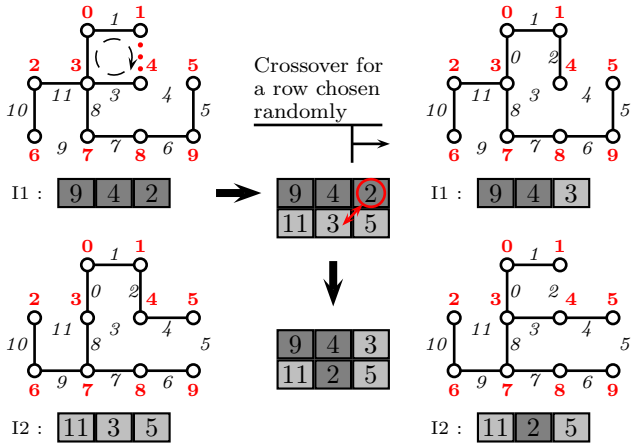


Figure 11: Crossover technique

5.4. Adjustments of GA parameters

[38] defines conventional GA parameters such as population size t_{pop}, crossover p_c and mutation p_m probabilities. These parameters have an influence on the GA performance. It is then necessary to study the impact of each probability on GA behavior to determine their optimal values. [29] proposes an approach allowing to achieve these adjustments. This method is applied in our research.

As a rough guide, note that a too small population would likely converge towards a minimum local. Conversely, a too large population increases the GA processing time and thereby it slows down the convergence time. In

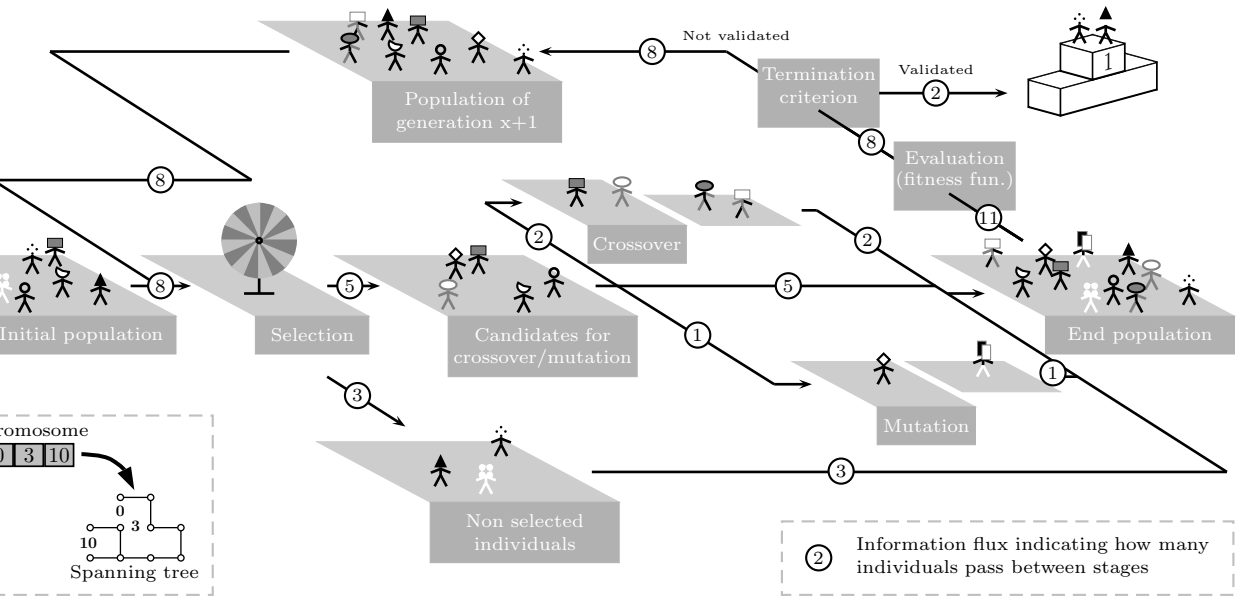
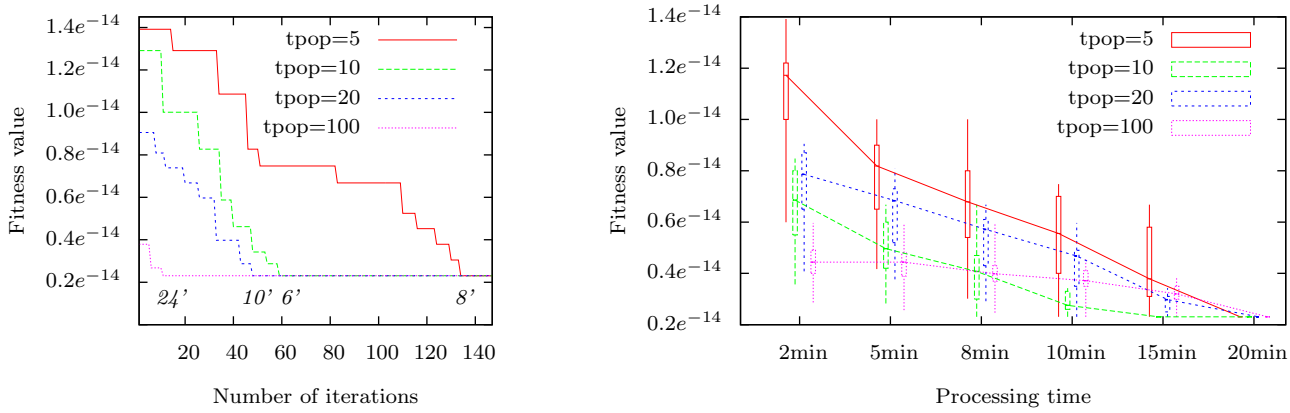


Figure 9: General principle of the natural selection



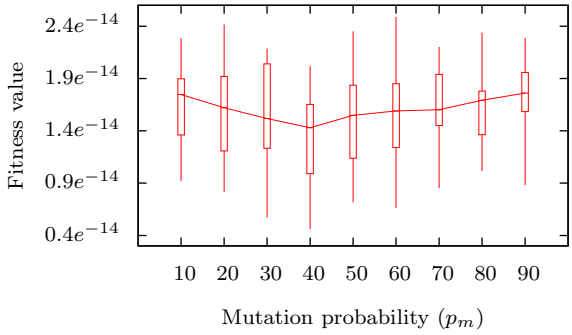
(a) Influence of t_{pop} on a single graph

(b) Influence of t_{pop} on the graph panel

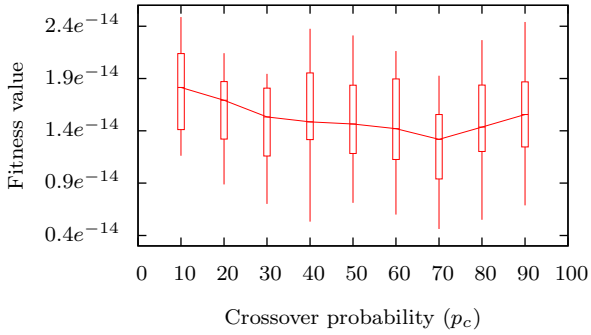
Figure 12: Adjustement of the t_{pop} parameter

conclusion, it is necessary to find a good balance between GA processing time and result quality. The population density depends on the coding, the used methods and the computing power. Four t_{pop} sizes are considered in our study (5, 10, 20 and 100). Two different views related to the t_{pop} influence are given in Figure 12. The first one (i.e. Figure 12(a)) depicts the behavior of GA when considering a unique run of a graph consisting of 100 nodes and 240 links. For each size, the evolution of the optimal value of the cost function according to the number of generations is drawn. The time spent to reach the same optimum is also given in Figure 12(a). A population size of 100 enables to obtain in a few generations the optimal solution, however its processing time is very long ($24min$) since GA performs a large number of crossover and mutation operations (due to the significant population size). A population of 5 shows GA converges in $8min$ but it needs a large number of iterations (approximately 140) due to the little population density/diversity. A population of 20 tends to reduce the processing time at $10min$ after 50 iterations. Nevertheless, the best arrangement for this specific graph is obtained with a population size of 10, since it finds the optimal solution in $6min$ after 60 iterations.

In addition, we synthesize the influence of t_{pop} in a box-and-whisker diagram (shown in Figure 12(b)) based on the 30 runs of a graph constituted of 81 nodes and 180 links. Figure 12(b) depicts the evolution of the optimal value of the cost function according to the processing time (and not the number of iterations). It is notable now that the convergence of GAs is not as pronounced as it is in Figure 12(a), the multiple runs tend to smooth the result curves. Indeed, the difference in time spent to reach



(a) Influence of p_m on the graph panel



(b) Influence of p_c on the graph panel

Figure 13: Adjustment of p_c and p_m parameters

a same optimum is not so significant. In the previous figure (Figure 12(a)), there were a maximum difference of $18min$ $(24' - 6')$ for reaching the same optimum between a population size of 10 and 100. Now, in Figure 12(b) there is a difference of $5min$ $(20' - 15')$ in average between the best solution (i.e. population size of 10) and the three others population sizes. Moreover, it is noticeable that the variation of the solutions (i.e between the 1^{st} and the 3^{th} quartiles) is getting smaller and smaller when the population size increases. If we look at each interval between the 1^{st} and the 3^{th} quartiles for each population size, we observe that this interval is the biggest for $t_{pop} = 5$ and the smallest for $t_{pop} = 100$. Indeed, the smaller the population, the less new individuals will be generated by iteration and therefore, the less the probability to have a new best solution. Let us note that according to the termination criterion, the population size could be tuned. Indeed, in case of time constrained applications, it may be sensible to choose a big population size (as 100) to get quickly a solution close to the optimal one. For instance, in Figure 12(b) a suitable solution for $t_{pop} = 100$ is reached in $2min$. In our case, we privilege the solution quality and Figure 12(b) shows that the GA converges faster toward this one for $t_{pop} = 10$ than the other population sizes, since the optimum is reached after approximatively 15 min versus $20min$ for the other sizes.

The diversification is introduced by the mutation operator p_m. This operator aims at avoiding convergence to local minima. In order to fix the mutation rate, GA is tested based on a panel of 100 graphs (with 150 iterations) for p_m varying from 10% to 90%, with $t_{pop} = 10$ and $p_c = 80\%$. We have chosen at random $p_c = 80\%$ knowing that usually, through the literature, the p_c value is upper to 60%. Figure 13(a) shows the fitness value relatively to the mutation rate. According to the diagram, the best mutation rate is around $p_m = 40\%$ since the five indicators given from the wisker-diagram (namely: the minimum, the 1^{st} and 3^{th} quartile, the mean and the maximum values) are lower for this probability than any other mutation rate.

The operator of crossover aims at enriching the population diversity. The more the crossover rate p_c is high, the more there are new individuals in the population tested. In order to fix the crossover rate, GA is executed for the same panel of graphs and the same number of iterations, for p_c varying from 10% to 90%, with $t_{pop} = 10$ and $p_m = 40\%$. Figure 13(b) shows the fitness values relatively to the crossover rates synthesized in a box-and-whisker diagram based on the graph panel. It suggests that a crossover rate around 70% gives the best results. Indeed, the lowest average fitness value is found for $p_c = 70\%$ which is equal to $1.3e^{-14}$. The comparison between the efficiency of the crossover probabilities is larger considering that 75% of results (3^{th} quartile) for $p_c = 70\%$ are inferior to $1.5e^{-14}$ contrary to the other p_c rates which vary approximatively from $1.8e^{-14}$ to $2.1e^{-14}$.

In resume, for the GA's parameters after a series of experiments are fixed to: $t_{pop} = 10$, $p_c = 70\%$ and $p_m = 40\%$, since the fitness function converges relatively quickly with a mutation probability lesser than the crossover one in order not to hinder effectiveness of the crossover exploration.

6. Case study

6.1. Introduction

Firstly, specific nodes were developed in the OPNET network simulation tool, to implement the behavior of NCS devices: Figure 14 illustrates the graphic interface of these new nodes tuning to the PLC cycle time, the communication cycle time, and the types of devices.

To give numerical results, the approach presented in this paper will be applied on a sample system with controller and process transfer functions $P(s)$ and $C(s)$ respectively such as:

$$P(s) = \frac{2}{(s+5)(s+0.2)}, C(s) = \frac{0.5508s + 0.4529}{s}$$

Secondly, the topology relies on a switched Ethernet architecture consisting of 31 switches and 56 links as depicted Figure 15. The NCS devices (controller, sensor,

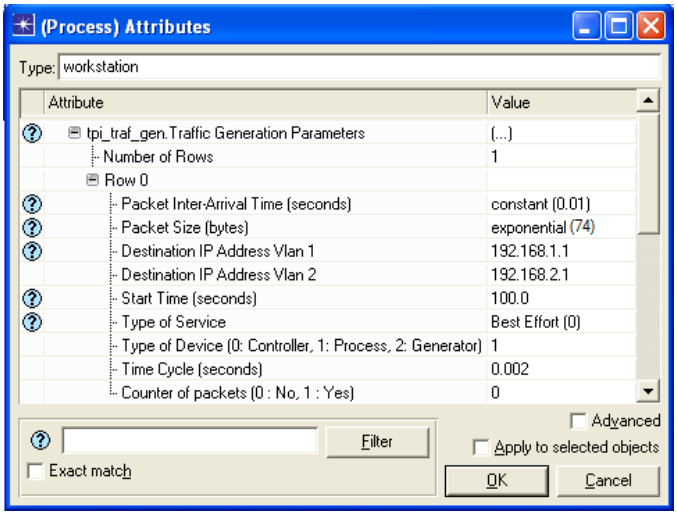


Figure 14: Opnet NCS node interface

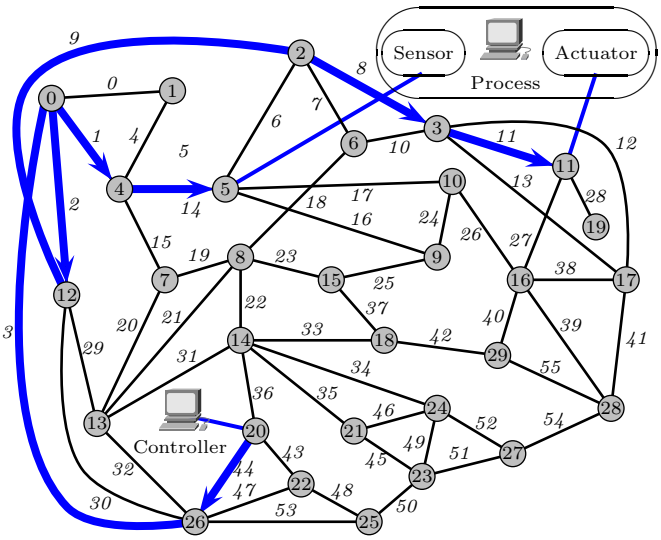


Figure 15: Spanning tree application

actuator) are respectively connected to switches 20, 5 and 11. The control sampling time is fixed at 2 ms whereas the measurements and control values (packets of 74 bytes) are exchanged each 1 ms.

Two cases are considered in this study. The first one uses the classic solution with the Spanning Tree Protocol (STP). The second one applies the method proposed in the present paper by implementing two paths obtained from GA. Rapid STP is activated on the simulation tool for defining both trees. To improve the understanding of the figures, only the links belonging to the paths between the NCS devices are highlighted (bold for the first tree and dashed bold lines for the second tree) and not the global tree.

6.2. NCS analysis with RSTP

Figure 15 shows the path used by the controller to communicate both with its sensor and actuator. This path is a part of one tree that can be defined by RSTP for a particular set of switches and ports' identifiers. During the simulation, the link 8, belonging to this path, fails at $t = 115$ s. Figure 17(a) provides the behavior of the process output according to the reference. Before the link failure, the process reaches the reference with an acceptable overshot. When the link 8 fails, the actuator is disconnected to the network and consequently to the controller during 5 s corresponding to the RSTP reconfiguration time period. Thus, between $t = 115$ s and $t = 120$ s, the new references are not taken into account by the process. After $t = 120$ s, the controller is able to send to the actuator the new control through a new path (resulting from the RSTP process), but an instability period is observed before this new control again reaches the reference. This instability step may be due to the reception of old messages (by the actuator) buffered in the network.

6.3. NCS analysis with two redundant paths relying on MSTP

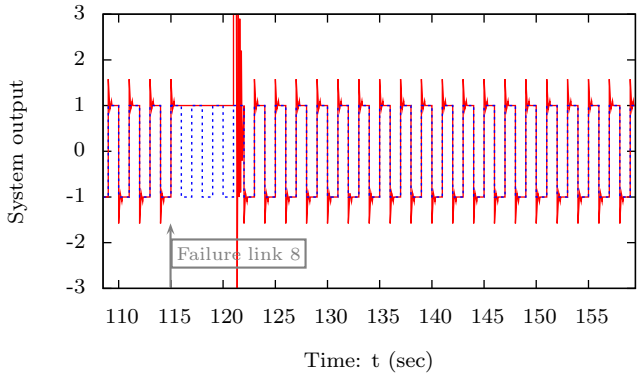
The redundant paths is estimated by using GA with the tuning defined in section 5.4: $t_{pop} = 10$, $p_c = 70\%$ and $p_m = 40\%$. The GA stops when an individual (pair of trees) reaches the SIL 4 level (i.e. the termination criterion, cf. Figure 9). Solutions for a small network as depicted Figure 15 were obtained in average after 2 minutes.

Figure 16 shows the result with two independent paths for interconnecting the controller and the process. Two VLANs are defined and two trees are created respectively for each VLAN by using MSTP. In practice, spanning trees will be configured by adapting the switches' ports priorities and the link costs. In this approach, the controller duplicates the control messages on both trees and it receives twice the sensor state.

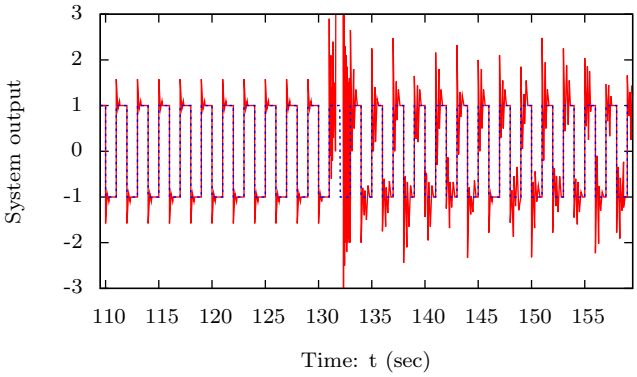
The sequence of link failures defined in the simulation is given by the following:

- *sequence n°1*: the link 6 fails at $t = 115s$ and is recovered at $t = 125s$,
- *sequence n°2*: the link 33 fails at $t = 135s$ and is recovered at $t = 145s$,
- *sequence n°3*: the two links 6 and 33 fail at $t = 160s$.

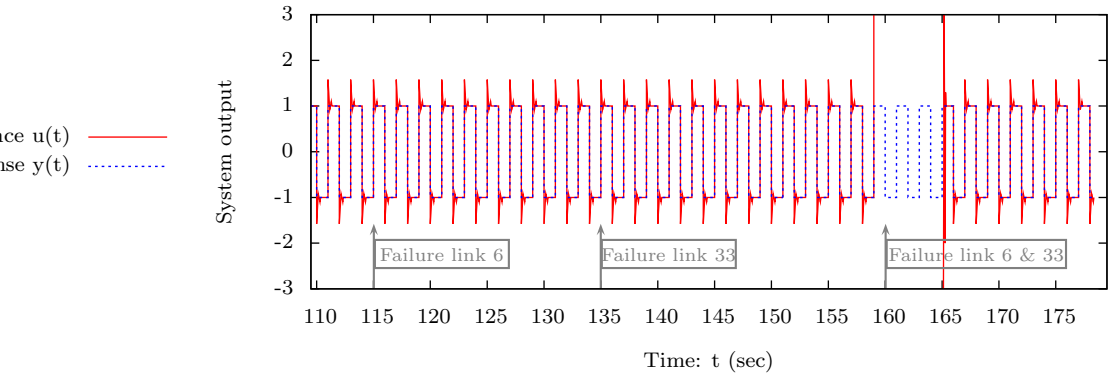
In sequence n°1, when the link 6 fails at $t = 120s$, Fig. 17(c) shows that the process continues to work because only the path associated to VLAN 1 is broken, and the connectivity is maintained by the second path associated to VLAN 2. The behavior is similar in sequence n°2 (i.e. when link 33 fails), but the paths are inverted (i.e. VLAN 2 is broken and not VLAN 1). For these two steps, the failure occurrences have no impact on the process control, as highlighted in Figure 17(c). In the last sequence (n°3), when both links 6 and 33 fail at $t = 160s$, the system becomes



(a) System output for the classic STP solution



(b) System output without packets numbering



(c) System output in case of duplicated messages

Figure 17: Simulation of the Network Control System behavior

unstable. Indeed, no message reaches the destinations anymore until a new path is recomputed by the STP algorithm (i.e. $5sec$ after the failure, namely $160 + 5sec$). However, this failure occurrence at $t = 160\ s$ appears with a probability at SIL 4 if all the network components are SIL 4 as it is explained in the section 4. In fact, we launched 30 runs (stopped when the fitness value stands for a SIL 4 solution) in order to obtain different pair of trees to interconnect the controller and the process. Note that no matter what the pair, the same behavior than in Figure 17(c) was observed. This shows that for this case study, GA enables to match a network performance level compatible with the NCS requirements. However, when failures occur simultaneously on both paths, performances of the communication will correspond to those provided by RSTP protocol.

6.4. Temporal validity of messages

The classical problem due to message duplication is the consistency control of the information system. The duplication generates desynchronizations since the duplicated messages do not arrive at the same time at the destination nodes. Figure 17(b) illustrates this problem by adding load traffic at $t = 130\ s$ on the link 42 (path VLAN 2). This overload generates congestion and some delays on messages which are forwarded across the path of VLAN 2. Consequently, the messages sent on path VLAN 1 arrive before the ones using the path VLAN 2. Figure 17(b) shows instabilities since the controller processes received messages in FIFO order, without managing their temporal validity. This trouble can be suppressed either by numbering or timing the messages. The numbering of messages is implemented in our OPNET simulations and the reception nodes discard all messages containing "old" numbers. This method is applied to eliminate the instabilities issues and the results are similar to the Figure 17(c).

7. Conclusion

The paper presents both a strategy to face the high dependability constraints and an off-line method to define, in practice the pair of paths for each traffic based on a local configuration (without modifying Ethernet standards). This method is based on two redundant paths determined and defined by a Genetic Algorithm and relies on message duplication incorporated in the MSTP technique. The main aim is to reduce the probability of communication disconnection in particular in a NCS framework. However, this method has to be applied only for specific nodes which are strongly timed constrained. It cannot be

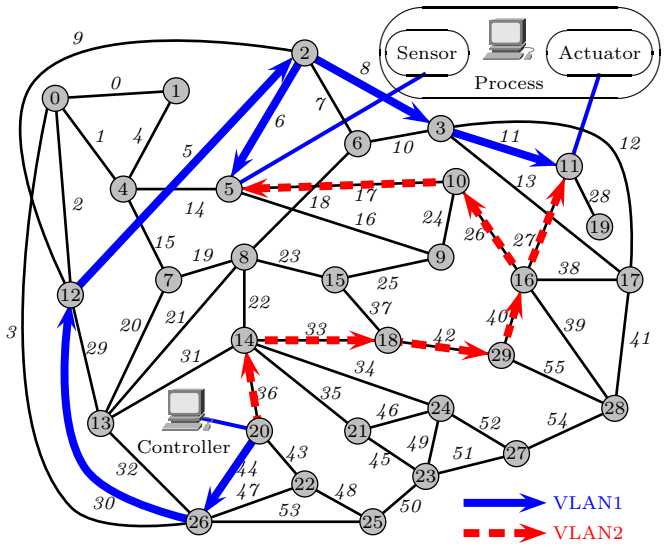


Figure 16: Message duplication on a Multi-VLAN architecture

generalized for all the nodes of the network, since the duplication of messages induces overload. Finally, current works are related to the application of this contribution for defining robust hybrid network infrastructures based on wireless access points.

Acknowledgement

This work was supported by OPNET Technologies, Inc. Thanks to "Teaching with OPNET" program.

References

[1] F. Lian, J. Moyne, D. Tilbury, Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet, Control Systems Magazine 21 (1) (2002) 66–83.

[2] J.-P. Georges, E. Rondeau, T. Divoux, Evaluation of switched Ethernet in an industrial context by using the network calculus, in: 4th International Workshop on Factory Communication Systems, 19–26, 2002.

[3] S. Zampieri, Trends in networked control systems, in: 17th IFAC World Congress, 6–11, 2008.

[4] W. Zhang, M. Branicky, S. Phillips, Stability of networked control systems, Control systems magazine 21 (1) (2001) 84–99.

[5] R. Anderson, M. Spong, Bilateral control of teleoperators with time delay, Transactions on Automatic Control 34 (5) (2002) 494–501.

[6] B. Hannaford, A design framework for teleoperators with kinesthetic feedback, Transactions on Robotics and Automation 5 (4) (2002) 426–434.

[7] F. Goktas, J. Smith, R. Bajcsy, Telerobotics over communication networks, 36th Conference on Decision and Control 3 (2002) 2399–2404.

[8] U. Ozguner, H. Goktas, H. Chan, J. Winkelman, M. Liubakka, R. Krtolica, Automotive suspension control through a computer communication network, in: 1st Conference on Control Applications, 895–900, 1992.

[9] F. Lian, J. Moyne, D. Tilbury, Implementation of networked machine tools in reconfigurable manufacturing systems, in: Proc. of the 2000 Japan-USA Symposium on Flexible Automation, Ann Arbor, MI, 2000.

[10] J. Hespanha, P. Naghshtabrizi, Y. Xu, A survey of recent results in networked control systems, Proceedings of the IEEE 95 (1) (2007) 138–162.

[11] K. Ji, W. Kim, Real-time control of networked control systems via Ethernet, International Journal of Control Automation and Systems 3 (4) (2005) 591–600.

[12] C. Lai, P. Hsu, Realization of networked control systems on Ethernet with varied time delay, in: International Conference on Systems Man and Cybernetics, SMC, 66–73, 2010.

[13] N. Vatanski, J.-P. Georges, C. Aubrun, E. Rondeau, S.-L. Jämsä-Jounela, Networked control with delay measurement and estimation, Control Engineering Practice 17 (2) (2009) 231–244.

[14] K. Lee, S. Lee, Performance evaluation of switched Ethernet for networked control systems, in: 28th Annual Conference of the Industrial Electronics Society, IECON, vol. 4, 3170–3175, 2003.

[15] R. Daoud, H. Elsayed, H. Amer, Gigabit Ethernet for redundant networked control systems, in: International Conference on Industrial Technology, ICIT'04, vol. 2, 869–873, 2004.

[16] B. Addad, S. Amari, Delay Evaluation and Compensation in Ethernet-Networked Control Systems, in: Giorgio Buttazzo, Pascale Minet (Eds.), 16th International Conference on Real-Time and Network Systems, RTNS, 2008.

[17] G. Jayavelu, S. Ramasubramanian, O. Younis, Maintaining colored trees for disjoint multipath routing under node failures, ACM Trans. Netw. 17 (1) (2009) 346–359.

[18] H. Kirrmann, K. Weber, O. Kleineberg, H. Weibel, HSR: Zero recovery time and low-cost redundancy for Industrial Ethernet (High availability seamless redundancy, IEC 62439-3), in: Conference on Emerging Technologies & Factory Automation, ETFA, 1–4, 2009.

[19] S. Limal, S. Potier, B. Denis, J.-J. Lesage, Formal verification of redundant media extension of Ethernet PowerLink, in: Conference on Emerging Technologies & Factory Automation, ETFA, 1045–1052, 2007.

[20] J. Imtiaz, J. Jasperneite, K. Weber, Redundant structures for a generic real-time ethernet system, in: Conference on Emerging Technologies & Factory Automation, ETFA, 1–4, 2010.

[21] D. Lee, K. Lee, S. Yoo, J. Rhee, Efficient Ethernet Ring Mesh Network Design, Journal of Lightwave Technology 29 (18) (2011) 2677–2683.

[22] J. Qiu, Y. Liu, G. Mohan, K. Chua, Fast spanning tree reconnection for resilient metro ethernet networks, in: Communications, 2009. ICC'09. IEEE International Conference on, 1–5, 2009.

[23] J. Qiu, G. Mohan, K. Chua, Y. Liu, Local restoration with multiple spanning trees in Metro Ethernet, in: Optical Network Design and Modeling, 2008. ONDM 2008. International Conference on, 1–6, 2008.

[24] T. Fencl, P. Burget, J. Bilek, Network topology design, Control Engineering Practice .

[25] M. Huynh, S. Goose, P. Mohapatra, Resilience technologies in Ethernet, Computer Networks 54 (1) (2010) 57–78.

[26] Y. Ali, S. Narasimhan, Sensor network design for maximizing reliability of linear processes, AIChE journal 39 (5) (1993) 820–828.

[27] J. Dréo, A. Petrowski, P. Siarry, E. Taillard, Metaheuristics for hard optimization : methods and case studies, Springer, 2006.

[28] B. Addad, S. Amari, J.-J. Lesage, Genetic algorithms for delays evaluation in networked automation systems, Engineering Applications of Artificial Intelligence 24 (3) (2011) 485–490.

[29] J.-P. Georges, N. Krommenacker, T. Divoux, E. Rondeau, A design process of switched Ethernet architectures according to real-time application constraints, Engineering Applications of Artificial Intelligence 19 (3) (2006) 335–344.

[30] B. Enacheanu, B. Raison, D. Ivanova, R. Caire, A. Aubry, N. Hadjsaid, Flexible Electric Infrastructures for Advanced Distribution Automation, in: Third International Conference on Critical Infrastructures, CRIS, Alexandria, US, 2006.

[31] K. Nara, A. Shiose, M. Kitagawa, T. Ishihara, Implementation of genetic algorithm for distribution systems loss minimum re-configuration, Transactions on Power Systems 7 (3) (1992)

1044–1051.
[32] Y. Zhu, K. Tomsovic, Adaptive Power Flow Method for Distribution Systems With Dispersed Generation, Power Engineering Review 22 (5) (2002) 72–72.
[33] B. Radha, R. King, H. Rughooputh, A modified genetic algorithm for optimal electrical distribution network reconfiguration, in: Congress on Evolutionary Computation, CEC, vol. 2, 1472–1479, 2004.
[34] E. R. Ramos, A. G. Exposito, J. R. Santos, F. L. Iborra, Path-based distribution network modeling: application to reconfiguration for loss reduction, IEEE Transactions on Power Systems 20 (3) (2005) 556–564.
[35] W.-M. Lin, F.-S. Cheng, M.-T. Tsay, Distribution feeder reconfiguration with refined genetic algorithm, Transactions on Power Systems Proceedings - Generation, Transmission and Distribution 147 (6) (2000) 349–354.
[36] Y.-Y. Hong, S.-Y. Ho, Determination of network configuration considering multiobjective in distribution systems using genetic algorithms, Transactions on Power Systems 20 (2) (2005) 1062 – 1069.
[37] B. Enacheanu, B. Raison, R. Caire, O. Devaux, W. Bienia, N. Hadjsaid, Radial Network Reconfiguration Using Genetic Algorithm Based on the Matroid Theory, Transactions on Power Systems 23 (1) (2008) 186–195.
[38] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.