Improving availability and safety of control systems by cooperation between intelligent transmitters

Florent Brissaud, Anne Barros, Christophe Bérenguer

Improving availability and safety of control systems by cooperation between intelligent transmitters

Florent Brissaud[a,b,*], Anne Barros[b], and Christophe Bérenguer[b]
[a]Institut National de l'Environement Industriel et des Risques (INERIS), Verneuil-en-Halatte, France
[b]Université de Technologie de Troyes (UTT), ICD, FRE CNRS 2848, Troyes, France
[*]florent.brissaud@ineris.fr, florent.brissaud@utt.fr

Abstract: "Intelligent transmitters" taking part in distributed and networked control systems are considered. Such sensor systems are able to perform internal data processing, advanced functionalities (e.g. self-diagnoses, online reconfiguration), and to exchange information. A control system made up of cooperating transmitters is therefore presented, using procedures which aim to improve system availability and safety. Taking advantage of the information exchanged between transmitters, two algorithms are proposed in order to i) retain the most confident values for transmitter processing, and ii) perform diagnoses by result comparisons. The control system is modelled by stochastic and coloured Petri nets, and dependability evaluations are performed by Monte Carlo simulations. Availability and safety can be put in balance through the use of the proposed algorithms, and both criteria can be improved under some conditions, notably according to diagnostic coverage of failures.

Keywords: Availability, Distributed Control Systems, Networked Control Systems, Intelligent Transmitters.

1. INTRODUCTION

Continuous progress in microelectronics and micromechanics is enabling systems to integrate more and more abilities into smaller and smaller packages. Industrial examples are the sensor systems: primarily used to collect data from the physical world, they are now also able to perform internal data processing and other functionalities like error measurement corrections, self-adjustment, self-diagnoses, online reconfigurations, and digital bidirectional communication [1]. These systems therefore appropriately refer to "transmitters" instead of "sensors," because the latter simply pertain to an ability to convert quantities. The use of such transmitters with "embedded intelligence" in control systems then allows the spatial relocation of some operations previously performed by a central controller, throughout the system, forming a distributed control system (DCS). In addition, in a networked control system (NCS), the elements of a DCS (transmitters, actuators, controllers, etc.) are interconnected by a real-time communication network [2]. Network protocols and fieldbuses for industrial control include Controller Area Network (CAN), Profibus, Foundation Fieldbus, and Device-Nets [3]. A special kind of NCS involves wireless sensor networks (WSN) [4, 5], but such systems introduce specific issues which are outside the scope of the present paper.

The defining feature of an NCS is that information (e.g. references, measurements, control signals) is exchanged using a network among control system elements (transmitters, actuators, controllers, etc.) [6, 7], by contrast with the traditional point-to-point connections. The use of a network also enables remote control and data transfers [7, 3]. NCS then provide many advantages, such as lower system complexity, reduced wiring cost, ease of maintenance, and flexibility, which bring these systems into several applications such as manufacturing, automobile, and aircraft [7]. On the other hand, drawbacks are the network-induced delays and packet dropouts, which may cause instability and affect NCS performance [8, 6, 9, 10, 3, 11]. The time delays come from the time sharing of the communication medium as well as the computation time required for communication processing [10, 9]. Packet dropouts are data lost while in transit through the network, typically resulting from transmission errors or buffer overflows [8]. Many research works therefore focus on NCS stability in order to guarantee constant transmission times, minimize or compensate delays, and analyse the effects on system performances [6, 9, 11].

In the context of industrial risk management, the most relevant NCS (or DCS) performance criteria relate to dependability. A generic framework on communication network dependability is provided by an international standard still under study [12]. An overview of the dependability constraints related to fieldbuses (problems of temporal coherence and traffic characteristics) can be found in the literature [13]. Corresponding failure modes and effects analyses (FMEA) have been also proposed [13]. Several other dependability analyses focus particularly on CAN protocol, for example to study the response time under transmission errors, using deterministic [14] or stochastic [15] models; or to study the effects of errors [16], using fault injection techniques. The electromagnetic interferences are the most common (transient) faults taken into account. Assuming both transient and permanent faults, a dependability evaluation of fieldbus networks has been also proposed [17], using stochastic Petri nets (SPN). With consideration for an application, the network behaviour under transmission errors has been likewise modelled by stochastic activity networks (SAN), a flexible extension of SPN, to evaluate NCS dependability [18]. The quantitative analyses are then performed by Monte Carlo simulations. Notice that, in these previous works, the focus is only given on the network, and failures of NCS (or DCS) elements such as transmitters and actuators are not assumed.

More global dependability analyses consider both the communication network and the NCS (or DCS) elements. For example, scenarios leading to NCS failures are analysed taking probabilities of message losses into account [19]. To this end, coloured Petri nets (CPN) followed by Monte Carlo simulations are used. However, except for the communication, the transmitters and actuators modelled in the latter work cannot really be described as "intelligent". For example, only two functions are assumed for a transmitter: to measure and to communicate; and the only failure mode consists in the transmission of an error message instead of the measurement, according to a constant probability. Similar limitations of the "intelligent features" of transmitters and actuators are considered in other non-Petri-based approaches for NCS (or DCS) dependability analyses [20, 21]. On the other hand, the reliability of "intelligent transmitters" has been analysed [22, 23], but without regard to networked systems, and relating interactions between control system elements are thus ignored.

Finally, NCS (or DCS) dependability studies can be found in the literature with respect to the fault tolerant control (FTC) strategies [24, 25], which aim at guaranteeing the system goal to be achieved in spite of faults, especially in order to improve system reliability and safety [26], and for example under cost constraints [27]. In addition, fault diagnoses (FD) [28] aim at detecting, isolating, and estimating the faults, which is sometimes required by FTC strategies. Notice that Petri nets are, here again, often used to model fault tolerant NCS (or DCS), for example by means of SPN [29] or SAN [26].

In the present paper, "intelligent transmitters" as elements of NCS (or DCS) are analysed from a dependability point of view. Special features of such systems, other than the particularity of communicating on a network, are taken into account. These characteristic, enabled by "embedded intelligence," consist in operations such as data processing, self-diagnoses and online reconfiguration. The contribution of this paper can therefore be appreciated as a link between reliability analyses of "intelligent transmitters" [22, 23], and previous works on NCS (or DCS) dependability [19]. Although the proposed model does not take the communication network dependability into account, some related works [18] can be easily integrated. The real-time communication is, however, used to make a network of transmitters which are allowed to implement operations by exchanging information. Such a control system made up of cooperating transmitters is introduced in Section 2, using procedures which aim at improving system availability and safety, based on two proposed algorithms. This control system is then modelled by stochastic and coloured Petri nets, and dependability evaluations are performed by Monte Carlo simulations in Section 3. Finally, the effects of the proposed algorithms on system availability and safety are discussed according to input parameters such as diagnostic coverage of failures.

2. CONTROL SYSTEM OF COOPERATING TRANSMITTERS ON A NETWORK

2.1. Basic Control System of Redundant Transmitters

A basic control system (CS), made up of three redundant transmitters, is firstly considered. That is, all three of the transmitters monitor the same quantities. An overview of this CS architecture can be seen in Figure 1, ignoring the communication network (in purple). Each transmitter involves both measurand (physical quantity to be measured) and influencing factor (quantities which impact the measurand evaluation) monitoring, using main and auxiliary transducers, respectively. The measurement result, as a function of measurand and influencing factors, is then performed by a processing unit. In addition, self-diagnoses are available for each transmitter element (main transducer, auxiliary transducer and processing unit), according to a given failure diagnostic coverage. Finally, the measurement result is transmitted to the CS main controller, using a communication interface. The signal transmission can be digital (e.g. using fieldbuses or wireless communication) or analogue (e.g. using the main communication standard in the process industry, that is, a 4-20 mA analogical signal). If at least one transmitter element failure is detected by self-diagnosis, an error signal (e.g. for an analogical signal: over 20 mA or under 4 mA) is transmitted instead of a signal which represents a measurement result. If at least one transmitter element failure has occurred but no failure has been detected by self-diagnosis, a signal which corresponds to a normal operation of equipment under control (e.g. non-detection of a hazardous event) is assumed to be transmitted; that is, only "dangerous" failures are assumed (and thus, no spurious mode). The CS main controller is then able to command a safety function to be performed by actuators (outside the scope of the present paper), if at least the majority of its non-error received signals provide an accurate measurement result. That is, the voting logic of the CS main controller is 2-out-of-3 if all of the transmitters are diagnosed as "operating" (according to their self-diagnoses), 1-out-of-2 if two and only two transmitters are diagnosed as "operating," 1-out-of-1 if one and only one transmitter is diagnosed as "operating," and finally, if no transmitter is diagnosed as "operating" then the CS provides a failure information.

Each transmitter therefore has three functional states: *operating*, if all of its elements are operating; *detected failure*, if at least one element failure has occurred and has been detected by self-diagnosis; and *undetected failure*, if at least one element failure has occurred and none has been detected by self-diagnosis. According to these functional states of the three transmitters, the functional state of the CS is summarised in Figure 2. Due to the CS voting logic introduced previously, the CS is in *operating* state if at least two transmitters are in operating state, or only one is in operating state but at least another one is in detected failure state; the CS is in *detected failure* state if all three of the transmitters are in detected failure state; and the CS is in *undetected failure* state in all other cases.

Figure 1: Control System Architecture

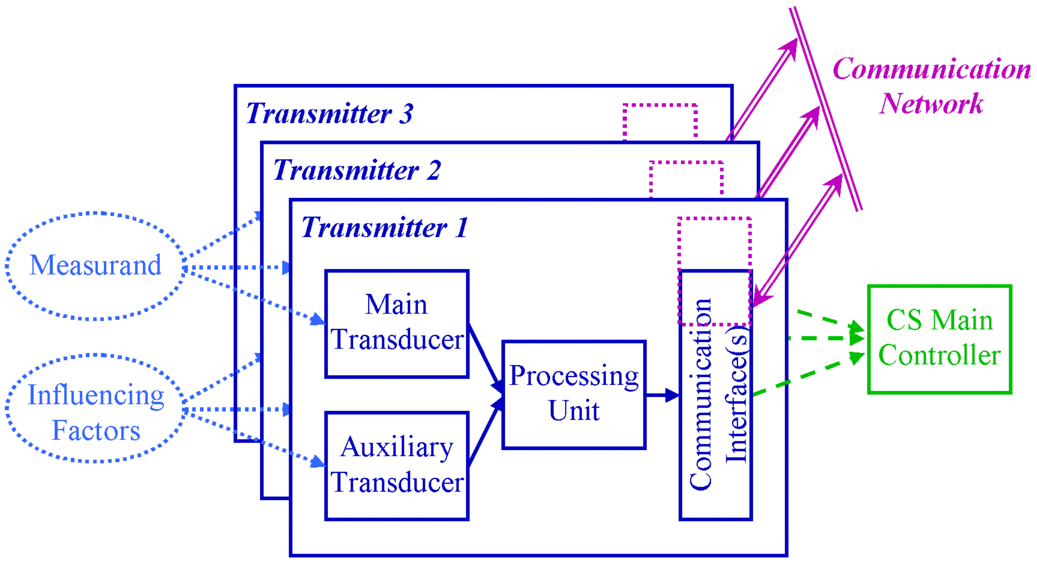
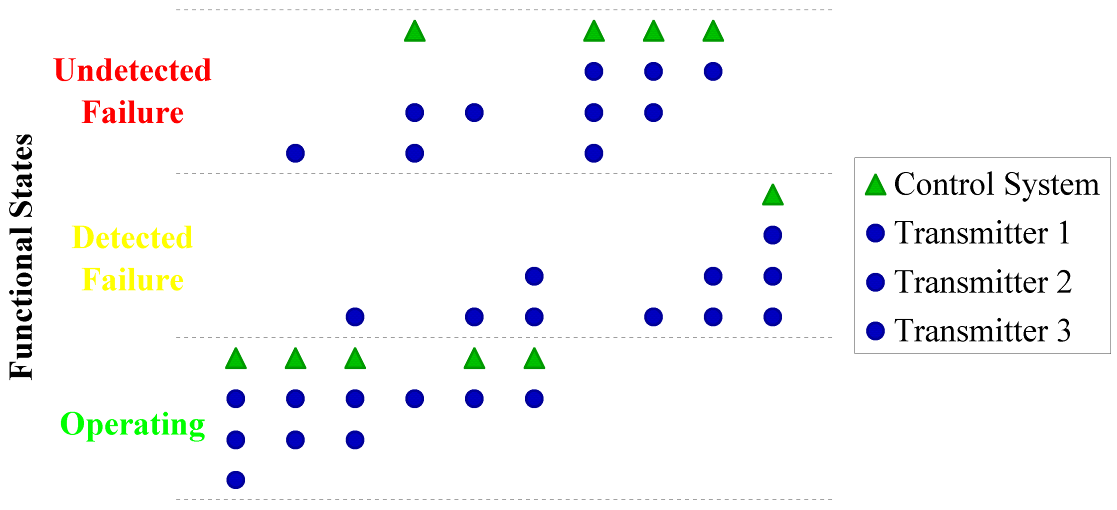


Figure 2: Control System Functional States



2.2. From Basic Control System to Networked Control System

For the present study, a communication network is assumed between the transmitters of the CS (in purple in Figure 1). This communication may, for example, be provided by an additional wireless communication interface, independently of the communication between each transmitter and the CS main controller. If the communication between the transmitters is not independent of the communication with the CS main controller, then specific issues have to be taken into account (cf. Section 1). This latter case is, however, not considered for the present study. The assumed communication module then allows the transmitters to exchange several kinds of information between them: transmitter identification, value of measurand (just as obtained in output from the main transducer), value of main transducer self-diagnosis, value of influencing factors (just as obtained in output from the auxiliary transducer), value of auxiliary transducer self-diagnosis, measurement result (obtained in output from the processing unit), value of processing unit self-diagnosis, and a diagnosis compilation result which is computed by the processing unit according to the three values of self-diagnoses. The result of this diagnosis compilation determines the signal transmitted to the CS main controller. It is either "confident," if the signal transmitted would represent the measurement result, or "not confident," if the signal transmitted would be an error signal.

A transmitter may be triggered according to two demand types. When a transmitter is triggered alone following a cyclic process (each transmitter is triggered in turn after a *delay* time), the corresponding demand type is denoted *cyclic*. When more than one transmitter is triggered at the same time, the corresponding demand type is denoted *parallel*. Because only one transmitter may use the communication network at once in order to prevent system instability (cf. Section 1), a transmitter is allowed to transmit information to the other transmitters only if it is triggered according to a *cyclic* demand type.

Alternately, each transmitter therefore performs its own evaluations (values of measurand and influencing factors, measurement result, values of self-diagnoses, diagnosis compilation result) according to a *cyclic* demand type, and transmits all of its information to the other transmitters of the network which, in turn (and after a *tempo* time), perform their own evaluations, according to a *parallel* demand type. Moreover, each time a transmitter is triggered (either according to a *cyclic* or *parallel* demand type), it transmits a signal to the CS main controller (either a signal which represents a measurement result or an error signal, according to the diagnosis compilation result). This demand process may be applied to CS made up of any number of transmitters. For the present study, the CS is made up of three transmitters, and the corresponding demand process is reported in Table 1.

Table 1: Demand Process for the Control System made up of Three Transmitters

Time	Demand	Signal or information transmitted
$t_0 = 0$	-	-
$t_1 = t_0$	*cyclic* demand of Transmitter 1	signal to the CS main controller information to the other transmitters
$t_2 = t_0 + tempo$*	*parallel* demand of Transmitter 2 *parallel* demand of Transmitter 3	signal to the CS main controller signal to the CS main controller
$t_3 = t_1 + delay$*	*cyclic* demand of Transmitter 2	signal to the CS main controller information to the other transmitters
$t_4 = t_2 + delay$	*parallel* demand of Transmitter 1 *parallel* demand of Transmitter 3	signal to the CS main controller signal to the CS main controller
$t_5 = t_3 + delay$	*cyclic* demand of Transmitter 3	signal to the CS main controller information to the other transmitters
$t_6 = t_4 + delay$	*parallel* demand of Transmitter 1 *parallel* demand of Transmitter 2	signal to the CS main controller signal to the CS main controller
$t_7 = t_5 + delay$	*cyclic* demand of Transmitter 1	signal to the CS main controller information to the other transmitters
$t_8 = t_6 + delay$	*parallel* demand of Transmitter 2 *parallel* demand of Transmitter 3	signal to the CS main controller signal to the CS main controller
etc.	etc.	etc.

**delay* is the time between two transmitter triggers according to a *cyclic* demand, and *tempo* < *delay*

2.3. Algorithms for Cooperating Transmitters

Based on two proposed algorithms, procedures are used to improve CS dependability by taking advantage of cooperation between transmitters. The two algorithms are denoted *Backup Algorithm* and *Contrast Algorithm*. The first refers to the measurand and influencing factor monitoring, performed by the main and auxiliary transducers. The second refers to the diagnosis compilation, performed by the processing unit. These algorithms are functionally independent and may or may not be used together. When a transmitter is triggered, it may therefore perform one or both of these algorithms. The algorithms use received information from other transmitters, and are based on self-diagnoses of transmitters' elements in order to compare, assess, or replace values used in transmitter processing. Finally, these algorithms can be performed regardless of the number of transmitters on the network.

The first algorithm, denoted *Backup Algorithm*, is used to replace a value which has been obtained for measurand (respectively for influencing factors) for the corresponding value from received information (originated from another transmitter triggered according to a *cyclic* demand type), if the latter is assumed more confident, according to the self-diagnoses of the main transducers (respectively the auxiliary transducers). In addition, a memorize process is used to select the assumed most confident values (for measurand and influencing factors), according to the self-diagnoses, among the last information received from other transmitters. The procedure based on *Backup Algorithm* is then as follows:

Backup Algorithm – When a transmitter is triggered:

Step 1: Perform transmitter evaluations (values of measurand and influencing factors, values of self-diagnoses of the main and auxiliary transducers), independently of the information received from other transmitters.

Step 2: Read the last information received from other transmitters.

Step 3: Read the last information memorized by the transmitter.

Step 4: Replace the last memorized information for the last received information, **if** the two transmitter identifications are the same, **or if** the values of measurand and/or influencing factors are assumed more confident, according to the corresponding self-diagnoses.

Step 5: The processing unit of the transmitter performs the measurement result by using the values of measurand and influencing factors which are assumed the most confident, according to the self-diagnoses, among the transmitter evaluations and the last memorized information.

Two versions of this algorithm are proposed: *Backup Algorithm* with *Algo 1* and *Backup Algorithm* with *Algo 2*. Using *Backup Algorithm* with *Algo 1*, the transmitter is able to replace values which have been obtained for measurand or influencing factors only if the demand type is *parallel* (that is, when no information is then transmitted to the other transmitters), in order to avoid that the transmitter performs *Backup Algorithm* by using information which can itself depend on the transmitter. That is, Step 5 of *Backup Algorithm* with *Algo 1* is performed only if the demand type is *parallel*. Using *Backup Algorithm* with *Algo 2*, the demand type is not considered and the previous situation is therefore not prevented. That is, Step 5 of *Backup Algorithm* with *Algo 2* is always performed.

The second algorithm, denoted *Contrast Algorithm*, is used to replace the result of the diagnosis compilation computed by the processing unit, for a diagnosis result obtained by comparisons of measurement results. When a transmitter is triggered according to a *parallel* demand type, it performs its own evaluations and then compares its measurement result to the corresponding value from received information (originated from another transmitter triggered according to a *cyclic* demand type). If two consecutive comparisons of measurement results, which are performed using information from different sending transmitters providing measurement results assumed confident according to the diagnosis compilation, have deduced that the measurement results are the same (respectively not the same), then the result of the diagnosis compilation is replaced for "confident" (respectively "not confident"). The procedure based on *Contrast Algorithm* is then as follows:

Contrast Algorithm – When a transmitter is triggered:

Step 1: Perform transmitter evaluations (measurement result, values of self-diagnoses, diagnosis compilation result), independently of the information received from other transmitters.

Step 2: Read the last information received from other transmitters.

Step 3: Compare the measurement result from the transmitter evaluations with the measurement result from the last received information, and memorize the transmitter identification from the last received information with a label which represents the match between the two measurement results ("positive match" if the results are the same, "negative match" otherwise), **if** the diagnosis compilation result from the last received information is "confident."

Step 4: Replace the diagnosis compilation result from the transmitter evaluations for "confident" (respectively "not confident"), **if** the two last memorized transmitter identifications with different names both have a "positive match" (respectively a "negative match") label, **and if** the demand type is *parallel*.

3. MODELLING AND EVALUATING COOPERATING TRANSMITTERS

3.1. Stochastic and Coloured Petri Nets

Petri nets are both graphical and mathematical tools. In particular, it is possible to set up state equations and mathematical models governing system behaviours [30]. An "ordinary" Petri net is made up of places (circles) and transitions (rectangles). Connections (directed arcs) may link a place to a transition (input arc) or vice-versa (output arc), and may be "valued" (otherwise the value is assumed to be one). These place-transition nets are therefore bipartite directed graphs. Places may contain tokens (small filled circles) which are "moved" through the enabled transitions when the latter are fired. A transition is enabled when each of its input places (linked to the transition by an input arc) contains a number of tokens equal to or greater than the corresponding input arc value. Firing an enabled transition then consists in two steps: first, removing, in each input place, a number of tokens equal to the corresponding input arc value; second, depositing, in each output place, a number of tokens equal to the corresponding output arc value. More details about these definitions and rules, examples, and Petri net properties may be found in the literature [31, 30].

Usually, the places of a Petri net represent objects or conditions, the tokens specify the values of these objects or conditions, and the transitions model the system activities. To allow the modelling of complex systems, several extended Petri nets have been developed, which notably include coloured, timed, and stochastic properties. In coloured Petri nets (CPN) [32], different types (colours) of tokens are represented in the same graph, improving the modelling facilities and clarity. More generally, labels (values) may be assigned to each token, and may be changed when firing transitions or staying in places (ageing tokens [33]). Firing policies may also depend on token labels. The time-dimension is introduced in timed Petri nets (TPN) by the use of sojourn times of tokens into places, and/or time delays for firing transitions. In stochastic Petri nets (SPN) [34], these delays are random variables. A flexible extension of SPN is the stochastic activity networks (SAN) [35]. When a Petri net includes both immediate and timed transitions, it is also denoted generalized Petri net (GPN) or generalized stochastic Petri net (GSPN).

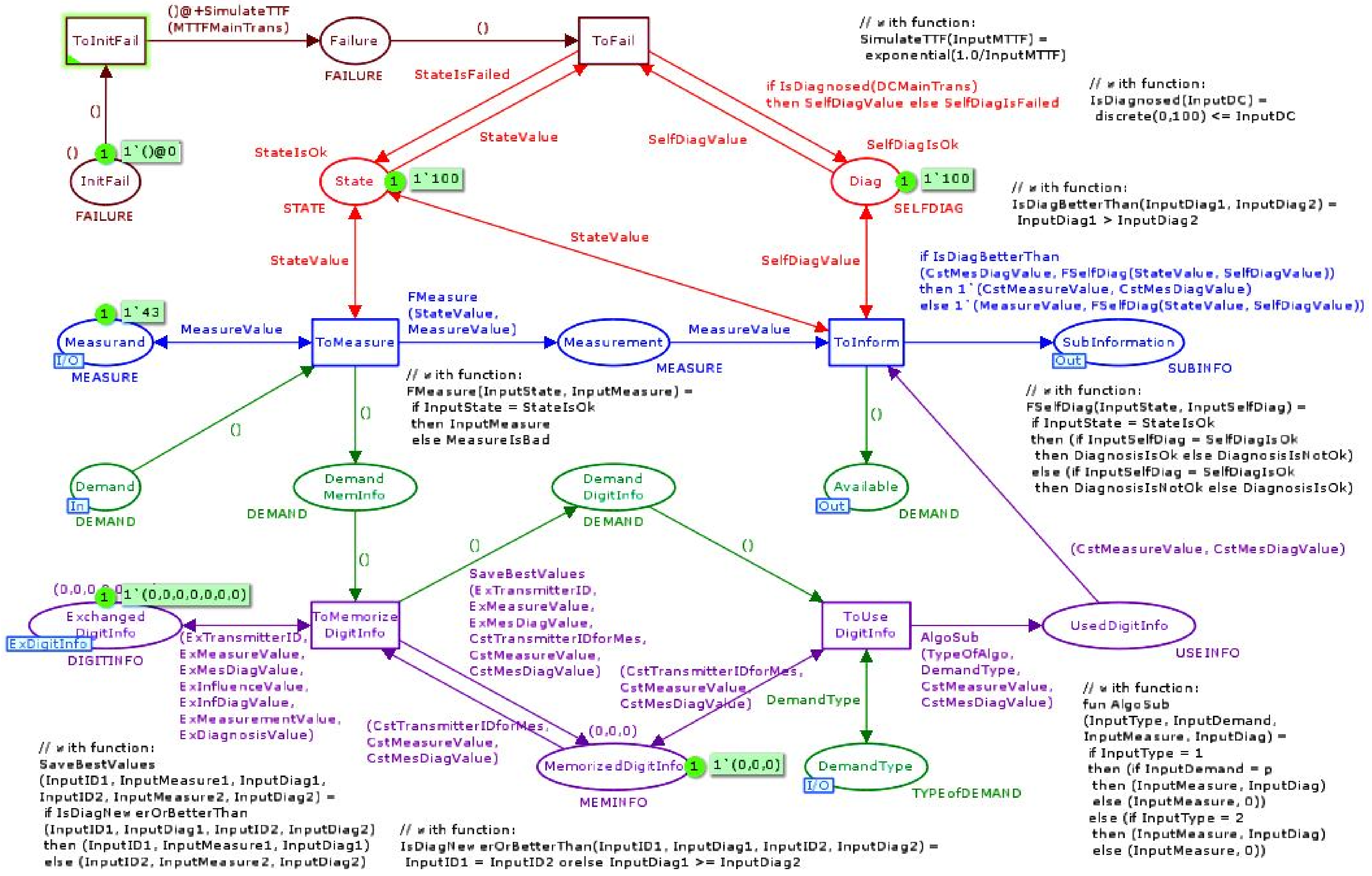
"Ordinary" and extended Petri nets then provide a very interesting tool for describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic and/or stochastic [30]. Nevertheless, a major weakness of Petri nets is the complexity problem: the more general the model, the less amenable it is to analysis [30]. For example, under some conditions, SPN may be analysed through the use of Markov Chains [34]. For more complex and generalized problems, the Petri nets may be directly used for qualitative analyses, but quantitative analyses often require Monte Carlo simulations which can be very time-consuming. Comprehensive information on Petri nets may, for example, be found on the Petri nets World website [36].

3.2. Modelling Framework

To model the CS by taking into account the cooperation between transmitters on the network, and then to assess dependability criteria, the stochastic and coloured Petri nets are claimed to be an intuitive and powerful tool. For example, Petri nets have already been used in several related studies [17, 18, 19, 29, 26] (cf. Section 1). While the stochastic properties are required to model random failures and imperfect self-diagnoses, the coloured characteristics provide a useful way to represent several kinds of information (e.g. values of measurand and influencing factors, measurement results, values of self-diagnoses, diagnosis compilation result) and properties (states of transmitter's elements, values' accuracy). CPN Tools [37] has then been chosen to model the CS. It is a free computer tool developed by the University of Aarhus for editing, simulating and analysing coloured Petri nets [37]. Among other interesting characteristics, CPN Tools supports coloured and both timed and untimed Petri nets. Although stochastic aspects are not directly available, the declaration possibilities for variables and expressions allow the inclusion of these properties [32]. In addition, hierarchical Petri nets can be created by assigning a separate Petri Net to a transition, or by creating fusion places, which is very interesting to model systems made up of any number of similar elements. On the practical side, the user interface, graphical editor, syntax checking and contextual error messages offer facilities for use [39]. Finally, to perform quantitative analyses, Monte Carlo simulations are required.

The stochastic and coloured Petri net for the main transducers is depicted in Figure 3. It is proposed as an example, and used for the modelling framework discussion. The Petri net for the auxiliary transducers is similar to the Petri net of Figure 3, except for some parameters and names of places, and it is therefore not depicted in another figure. Other Petri nets have also been designed to model the processing units, the whole transmitters (which hierarchically includes Petri nets for main transducers, auxiliary transducers and processing units, plus the communication interfaces), and to model the CS (which hierarchically includes three Petri nets for whole transmitters), but are not depicted in the present paper. The hierarchy between Petri nets is given by labels attached to the bottom left of places (sky blue rectangles in Figure 3). For example, the "out" label on the "SubInformation" place represents an output link, the "in" label on the "Demand" place represents an input link, and both are associated with places of another Petri net. "I/O" labels represent both input and output links, and other labels such as "ExDigitInfo" are assigned to places which are common to several Petri nets (fusion places).

Figure 3: Stochastic and Coloured Petri Net for Main Transducers


ToInitFail
()@+SimulateTTF (MTTFMainTrans)
Failure
FAILURE
()
ToFail
// with function: SimulateTTF(InputMTTF) = exponential(1.0/InputMTTF)
// with function: IsDiagnosed(InputDC) = discrete(0,100) <= InputDC
StateIsFailed
if IsDiagnosed(DCMainTrans) then SelfDiagValue else SelfDiagIsFailed
()
1`()@0
InitFail
FAILURE
StateIsOk
StateValue
SelfDiagValue
SelfDiagIsOk
State
1`100
STATE
Diag
1`100
SELFDIAG
// with function: IsDiagBetterThan(InputDiag1, InputDiag2) = InputDiag1 > InputDiag2
StateValue
StateValue
SelfDiagValue
if IsDiagBetterThan (CstMesDiagValue, FSelfDiag(StateValue, SelfDiagValue)) then 1`(CstMeasureValue, CstMesDiagValue) else 1`(MeasureValue, FSelfDiag(StateValue, SelfDiagValue))
1`43
Measurand
I/O
MEASURE
MeasureValue
ToMeasure
FMeasure (StateValue, MeasureValue)
Measurement
MEASURE
MeasureValue
ToInform
SubInformation
Out
SUBINFO
// with function: FMeasure(InputState, InputMeasure) = if InputState = StateIsOk then InputMeasure else MeasureIsBad
// with function: FSelfDiag(InputState, InputSelfDiag) = if InputState = StateIsOk then (if InputSelfDiag = SelfDiagIsOk then DiagnosisIsOk else DiagnosisIsNotOk) else (if InputSelfDiag = SelfDiagIsOk then DiagnosisIsNotOk else DiagnosisIsOk)
Demand
In
DEMAND
Demand MemInfo
DEMAND
Demand DigitInfo
DEMAND
Available
Out
DEMAND
(CstMeasureValue, CstMesDiagValue)
(0,0,0,0,0,0,0,0)
1`(0,0,0,0,0,0,0,0)
Exchanged DigitInfo
ExDigitInfo
DIGITINFO
(ExTransmitterID, ExMeasureValue, ExMesDiagValue, ExInfluenceValue, ExInfDiagValue, ExMeasurementValue, ExDiagnosisValue)
ToMemorize DigitInfo
SaveBestValues (ExTransmitterID, ExMeasureValue, ExMesDiagValue, CstTransmitterIDforMes, CstMeasureValue, CstMesDiagValue)
(CstTransmitterIDforMes, CstMeasureValue, CstMesDiagValue)
(CstTransmitterIDforMes, CstMeasureValue, CstMesDiagValue)
(0,0,0)
MemorizedDigitInfo
1`(0,0,0)
MEMINFO
DemandType
ToUse DigitInfo
AlgoSub (TypeOfAlgo, DemandType, CstMeasureValue, CstMesDiagValue)
UsedDigitInfo
USEINFO
DemandType
I/O
TYPEofDEMAND
// with function: fun AlgoSub (InputType, InputDemand, InputMeasure, InputDiag) = if InputType = 1 then (if InputDemand = p then (InputMeasure, InputDiag) else (InputMeasure, 0)) else (if InputType = 2 then (InputMeasure, InputDiag) else (InputMeasure, 0))
// with function: SaveBestValues (InputID1, InputMeasure1, InputDiag1, InputID2, InputMeasure2, InputDiag2) = if IsDiagNewerOrBetterThan (InputID1, InputDiag1, InputID2, InputDiag2) then (InputID1, InputMeasure1, InputDiag1) else (InputID2, InputMeasure2, InputDiag2)
// with function: IsDiagNewerOrBetterThan(InputID1, InputDiag1, InputID2, InputDiag2) = InputID1 = InputID2 orelse InputDiag1 >= InputDiag2

In Figures 3, the measure chain is in blue (the tokens represent values for measurand, and are next associated with values of self-diagnoses); the states of the transducer elements and self-diagnoses are in red (tokens represent “ok” or “failed” states, and “diagnosed” or “undiagnosed” failure); the (random) failure aspects are in brown (a failure occurs when the “ToFail” transition is fired, changes the element states, and is detected by self-diagnosis according to a failure diagnostic coverage); the management of the demand process is in green (tokens represent either the demand occurrences for elements, or the demand types); the exchanged information between transmitters and the management of the procedure based on *Backup Algorithm* (the only relevant algorithm for Figure 3) are in purple (tokens represent values of exchanged information, and variables used by the algorithm).

The number of tokens in a non-empty place is given in an attached green filled circle, and the token values are given in the green rectangle close to it. The notation “*1`*” specifies a number of one token, and may be extended to any number. A place is able to contain only one kind of token defined by the colour type specified in capital letters, below each place. The initial number of tokens in each place is defined by the initial token values in lower-case, above each place. For example, the “State” place in Figure 3 contains one token of colour type “STATE” (which corresponds to integer numbers), and the token value is “*100*”, which corresponds to the initial token value “*StateIsOk*”. Other colour types may be “uncoloured”, for example the “InitFail” places contain tokens of type “FAILURE” which are only represented by the symbol “()”; may be defined by a subset, for example the “DemandType” places contain tokens of type “TYPEofDEMAND” which may have either “*c*” value (associated with *cyclic* transmitter demand type) or “*p*” value (associated with *parallel* transmitter demand type); or may be associated with multiple variables, for example the “Exchanged DigitInfo” places contain tokens of type “DIGITINFO” which groups together seven integer variables. When a colour type is timed, the corresponding token values are given with the symbol “@” followed by an integer value which represents the instant in time the token becomes available (before this instant in time, a token cannot be handled, that is, it cannot be used to fire transition).

An input arc (from a place to a transition) specifies the number of tokens (“*1`*” by default) to remove in the input place, with an input variable which takes the value of the removed token. An output arc specifies the number of tokens to deposit in the output place, with an output variable which gives the value to the deposited token. This latter variable may be a function of input variables (these functions are reported in black in Figures 3). For example, the token which is deposited in the “Measurement” place in Figure 3 takes the value of the input variable “*MeasureValue*” if the input variable “*StateValue*” is equal to “*StateIsOk*”, and the value “*MeasureIsBad*” otherwise. Using bi-directed arcs (e.g. between the “Measurand” place and the “ToMeasure” transition), the token is unchanged in the input place after firing the transition, but the token value may, however, be used as input variable in functions for output arcs. For timed tokens, the time delay may also be changed by output arcs. For example, when firing the transition “ToIntiFail”, the token deposited in the “Failure” place has a time delay increased by a random variable which follows an exponential distribution of mean equal to “*MTTFMainTrans*”. Input parameters used for random functions are given in Table 2. Notice that a cold standby has been assumed for the processing units of each transmitter. That is, when the first failure of a processing unit occurs, there is a probability equal to “*DCProcUnitStBy*” that it still operates, for a time which follows an exponential distribution of mean equal to “*MTTFProcUnitStBy*.”

Table 2: Input Parameters for Modelling and Evaluating the Control System

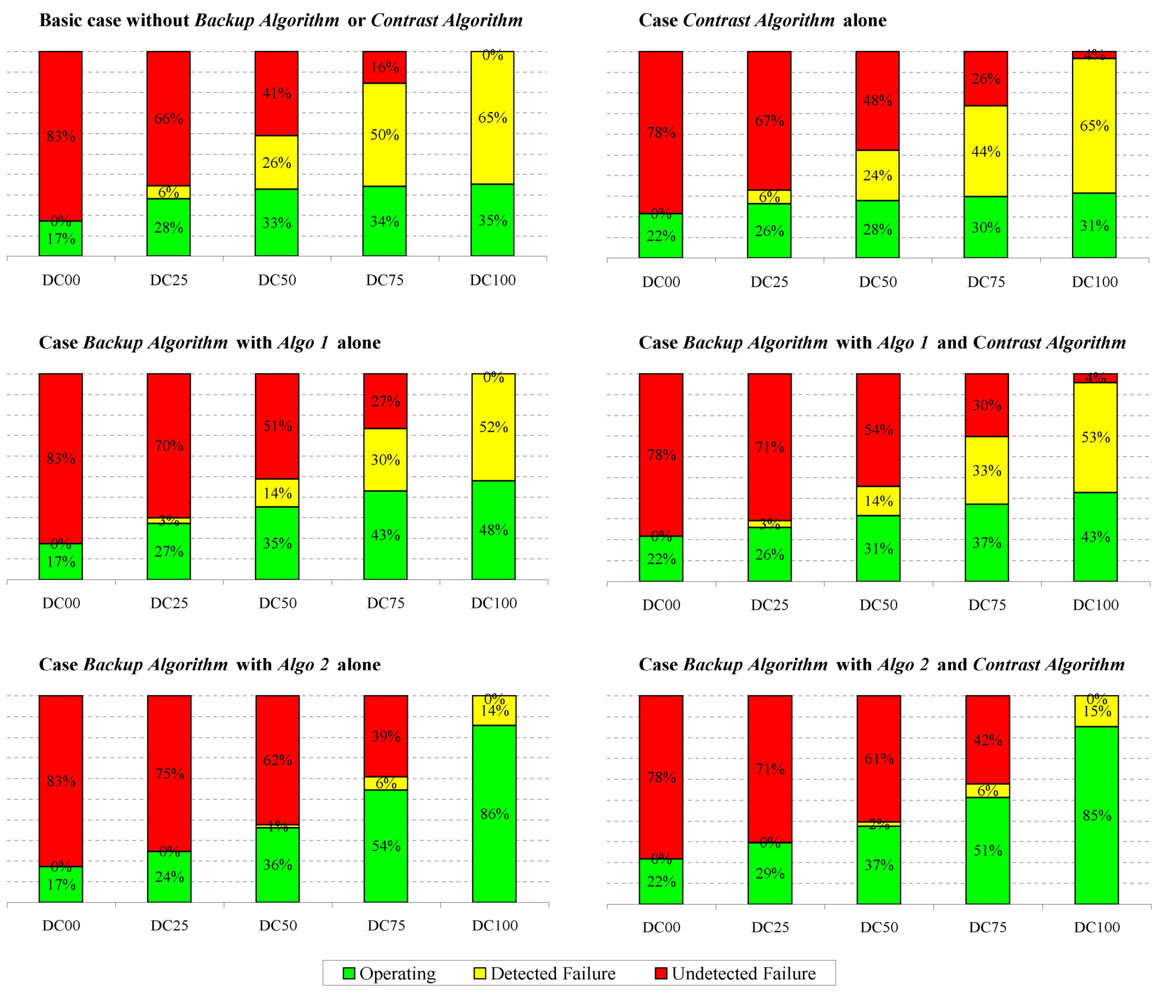
Name	**Description**	**Values**
MTTFMainTrans	Mean times to failure of main transducers, auxiliary transducers, and processing units, respectively, used for failure simulations according to exponential distributions	*5.000* time units
MTTFAuxTrans		
MTTFProcUnit		
MTTFProcUnitStBy	Mean times to failure of processing units in cold standby	*2.500* time units
DCMainTrans	Failure diagnostic coverage of main transducers, auxiliary transducers, and processing units, respectively, that is, probabilities to detect a failure when it occurs	*0.00*, *0.25*, *0.50*, *0.75*, or *1.00*
DCAuxTrans		
DCProcUnit		
DCProcUnitStBy	Probabilities of success of processing unit cold standby	*0.75*

3.3. Availability and Safety Analyses

The times to failure of each transmitter element (main transducers, auxiliary transducers, processing units) follow exponential distributions with means specified in Table 2 (in time units). Since Monte Carlo simulations are used for the analyses, any other distribution can also be used. When a failure of an element occurs, it is detected by self-diagnosis according to a constant probability equal to the diagnostic coverage. The failure diagnostic coverage is assumed to be the same for each transmitter element, and five values are used in order to analyse the impact of this parameter on the CS dependability, and according to the use of the proposed algorithms. Because no maintenance action is assumed, the criterion used to assess the CS availability and safety is the percentage of time that the CS spends in each functional state during the first 10,000 time units. The availability is then computed as the probability (percentage of time) that the CS is in operating state, and the safety is computed as the probability (percentage of time) that the CS is not in undetected failure state.

Six cases are analysed, according to the use (or not) of *Backup Algorithm* (with *Algo 1* or *Algo 2*) and/or *Contrast Algorithm* (cf. Section 2). For each case, different failure diagnostic coverage of each transmitter element is used: all diagnostic coverage equal to "*0.00*" in the "DC00" configurations, all equal to "*0.25*" in the "DC25" configurations, and etcetera up to "*1.00*" in the "DC100" configurations. Evaluations have been performed by 500 draws of Monte Carlo simulations for each configuration, which provides results with an estimated 95% confidence interval at plus or minus 3% around the averages. The average results obtained for each configuration are reported in Figure 4.

Figure 4: Average Percentage of Time the Control System spends in each Functional State, during the Fist 10,000 Time Units



4. DISCUSSION OF RESULTS AND CONCLUSION

According to the results reported in Figure 4, the following remarks can be made:

▪ *Backup Algorithm* globally increases the CS availability, but decreases the CS safety.

▪ These previous gaps are increasing according to the failure diagnostic coverage, and are even more significant through the use of *Backup Algorithm* with *Algo 2* than with *Algo 1*.

▪ *Contrast Algorithm* globally increases both availability and safety when the failure diagnostic coverage is very low, but decreases both of them when the failure diagnostic coverage is greater.

▪ The diagnostic coverage threshold between these previous effects of *Contrast Algorithm* is greater through the use of *Backup Algorithm* with *Algo 2* than with *Algo 1*.

When using *Backup Algorithm*, an element which is diagnosed as “failed” may be functionally replaced by another element which is diagnosed as “operating.” Nevertheless, an element diagnosed as “operating” may, in fact, be “operating,” but may also be “undetected failed.” Therefore, although *Backup Algorithm* increases availability (when an unavailable function becomes available), it also decreases safety (when an unavailable function, diagnosed as such, stays unavailable, but undiagnosed as such), and especially when the failure diagnostic coverage is low. On the other hand, when the failure diagnostic coverage is high, the diagnosis obtained by the use of *Contrast Algorithm* is less confident that the diagnosis compilation based on self-diagnoses. It is therefore recommended that:

▪ The use of *Backup Algorithm* should depend on the balance between availability and safety.

▪ The use of *Backup Algorithm* with *Algo 2* instead of *Algo 1* could be more justified when the failure diagnostic coverage is very high.

▪ *Contrast Algorithm* should be used when the failure diagnostic coverage is very low, or a bit less low according to the use of *Backup Algorithm*.

By using coloured and stochastic Petri nets, it has been possible to model a CS made up of “intelligent transmitters” and then to evaluate dependability criteria using Monte Carlo simulations. Two algorithms, namely *Backup Algorithm* and *Contrast Algorithm*, have been proposed in order to take advantage of the cooperation between transmitters to improve CS dependability. According to the percentage of time that the CS spends in each functional state during 10,000 time units, it has been shown that the CS availability and safety can be improved under some conditions (notably, according to the failure diagnostic coverage of transmitter elements) and, at least, can be put in balance by the use of the proposed algorithms. As a main conclusion, the results of the present study provide an indication that the new technologies inside transmitters, when appropriately used, not only provide practical advantages but may also be used to improve dependability criteria.

Acknowledgements

We would like to thank the French Ministry of Higher Education and Research, and the French Ministry of Ecology, Energy, and Sustainable Development for supporting our research work.

References

[1] F. Brissaud, A. Barros, C. Bérenguer and D. Charpentier. “*Dependability Issues for Intelligent Transmitters and Reliability Pattern Proposal*,” Proceedings of the 13th IFAC Symposium on Information Control Problems in Manufacturing, Moscow, Russia, (2009).

[2] F.-Y. Wang and D. Liu. “*Networked Control Systems – Theory and Applications*,” Springer-Verlag, London, (2008).

[3] Y. Tipsuwan and M.-Y Chow. “*Control methodologies in networked control systems*,” Control Engineering Practice, vol. 11, pp. 1099-1111, (2003).

[4] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci. “*A survey on sensor networks*,” IEEE Communication Magazine, vol. 40, pp. 104-112, (2002).

[5] J. Yick, B. Mukherjee and D. Ghosal. “*Wireless sensor network survey*,” Computer Networks, vol. 52, pp. 2292-2330, (2008).

[6] B.W. Zhang, M.S. Branicky and S.M. Philips. “*Stability of Networked Control Systems*,” IEEE Control Systems Magazine, vol. 21, pp. 84-99, (2001).

[7] R.A. Gupta and M.-Y. Chow. “*Overview of Networked Control Systems*,” Book chapter in Networked Control Systems – Theory and Applications, Springer-Verlag, London, (2008).

[8] J.P. Hespanha, P. Naghshtabrizi and Y. Xu. "*A Survey of Recent Results in Networked Control Systems*," Proceedings of the IEEE, vol. 95, pp. 138-162, (2007).
[9] Y. Ge, L. Tian and Z. Liu. "*Survey on the stability of networked control systems*," Journal of Control Theory and Applications, vol. 5, pp. 374-379, (2007).
[10] F.-L. Lian, J. Moyne and D. Tilbury. "*Network Design Consideration for Distributed Control Systems*," IEEE Transactions on Control Systems Technology, vol. 10, pp. 297-307, (2002).
[11] M.S. Branicky, S.M. Philips and W. Zhang. "*Stability of Networked Control Systems: Explicit Analysis of Delay*," Proceedings of the American Control Conference, Chicago, USA, (2000).
[12] International Electrotechnical Commission. "*IEC 61907 Ed. 1.0: Communication Network Dependability Engineering*," IEC, Geneva, 2009.
[13] L. Cauffriez, J. Ciccotelli, B. Conrard, M. Bayart and the members of the working-group CIAME. "*Design of intelligent distributed control systems: a dependability point of view*," Reliability Engineering and System Safety, vol. 84, pp. 19-32, (2004).
[14] K. Tindell, A. Burns and A.J. Wellings. "*Calculating controller area network (CAN) message response times*," Control Engineering Practice, vol. 3, pp. 1163-1169, (1995).
[15] N. Navet, Y.-Q. Song, F. Simonot. "*Worst-case deadline failure probability in real-time applications distributed over controller area network*," Journal of Systems architecture, vol. 46, pp. 607-617, (2000).
[16] J. Pérez, M.S. Reorda and M. Violante. "*Stability of Networked Control Systems: Explicit Analysis of Delay*," Proceedings of the 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, Boston, USA, (2003).
[17] P. Portugal and A. Carvalho. "*A framework for dependability evaluation of fieldbus networks*," Proceedings of the 5th International Workshop on Factory Communication Systems, Vienna, Austria, (2004).
[18] R. Ghostine, J.-M. Thiriet, J.-F. Aubry. "*Dependability evaluation of networked control systems under transmission faults*," Proceedings of the 6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Beijing, China, (2006).
[19] P. Barger, J.-M. Thiriet and M. Robert. "*Safety analysis and reliability estimation of a networked control system*," Proceedings of the 5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Washington, USA, (2003).
[20] Y.S. Dai, M. Xie, K.L. Poh and G.Q. Liu. "*A study of service reliability and availability for distributed systems*," Reliability Engineering and System Safety, vol. 79, pp. 103-112, (2003).
[21] B. Conrard, J.-M. Thiriet and M. Robert. "*Distributed system design based on dependability evaluation: a case study on a pilot thermal process*," Reliability Engineering and System Safety, vol. 88, pp. 109-119, (2005).
[22] F. Brissaud, A. Barros, C. Bérenguer and D. Charpentier. "*Reliability Study of an Intelligent Transmitter*," Proceedings of the 14th ISSAT International Conference on Reliability and Quality in Design, San Francisco, USA, (2009).
[23] F. Brissaud, A. Barros, C. Bérenguer and D. Charpentier. "*Reliability analysis for new technology-based transmitter*," Submitted for publication (2010).
[24] N.H. El-Farra, A. Gani and P.D. Christofides. "*Fault-Tolerant Control of Process Systems Using Communication Networks*," AIChE Journal, vol. 51, pp. 1665-1682, (2005).
[25] Z. Mao and B. Jiang. "*Fault identification and fault-tolerant control for a class of networked control systems*," International Journal of Innovative Computing, Information and Control, vol. 3, pp. 1121-1130, (2007).
[26] J.C. Campelo, P. Yuste, F. Rodriguez, P.J. Gil and J.J. Serrano. "*Hierarchical reliability and safety models of fault tolerant distributed industrial control systems*," Proceedings of the 18th International Conference on Computer Safety, Reliability and Security, Toulouse, France, (1999).
[27] N. Wattanapongsakorn and S. Levitan. "*Reliability Optimization Models for Fault-Tolerant Distributed Systems*," Proceedings of the Annual Reliability and Maintainability Symposium, Philadelphia, USA, (2001).
[28] H. Fang, H. Ye and M. Zhong. "*Fault diagnosis of networked control systems*," Annual Reviews in Control, vol. 31, pp. 55-68, (2007).
[29] J.R. Pimentel and M. Salazar. "*Dependability of Distributed Control System Fault Tolerant Units*," Proceedings of the 28th Annual Conference of the IEEE Industrial Electronics Society, Seville, Spain, (2002).
[30] T. Murata. "*Petri Nets: Properties, Analysis and Applications*," Proceedings of the IEEE, vol. 77, pp. 541-580, (1989).
[31] J.L. Peterson. "*Petri Net Theory and the Modeling of Systems*," Prentice-Hall, New York, (1981).
[32] K. Jensen. "*Coloured Petri Nets: basic concepts, analysis methods and practical use*," Springer-Verlag, London, 2nd edition, (2002).
[33] V. Volovoi. "*Modeling of system reliability Petri nets with aging tokens*," Reliability Engineering and System Safety, vol. 84, pp. 149-161, (2004).
[34] F. Bause, P.S. Kritzinger. "*Stochastic Petri Nets – An introduction to the theory*," Vieweg Verlag, Germany, 2nd edition, (2002).
[35] J.F. Meyer, A. Movaghar, W.H. Sanders. "*Stochastic activity networks: structure, behaviour and application*," Proceedings of the International Conference on Timed Petri Nets, Torino, Italy, (1985).
[36] Petri Nets World. *http://www.informatik.uni-hamburg.de/TGI/PetriNets/*
[37] CPN Tools. *http://wiki.daimi.au.dk/cpntools/*
[38] K. Jensen, L.M. Kristensen, L. Wells. "*Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems*," International Journal on Software Tools for Technology Transfer, vol. 9, pp. 213-254, (2007).
[39] A.V. Ratzer, L. Wells, H.M. Lassen, *et al*. "*CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets*," Proceedings of the 24th International Conference on the Application and Theory of Petri Nets, Eindhoven, The Netherlands, (2003).