Testing of Agreement Procedure by Fault Injection

Taschko Nikolov[1]

Abstract – **This paper presents a fault injection method to test the correctness of agreement procedure. Unlike other injectors the faults are derived automatically from an attributed Petri net model. Paths in the reachability graph of the model comprise path classes characterized by time and data attributes, which are processed symbolically. Experimental results demonstrate the feasibility of the method.**

Keywords – **modeling, testing, fault injection, attributed Petri net**

I. INTRODUCTION

In many cases two or more nodes have to agree a common action to environment (such as agreement for the new routing table in telecommunication network, control system with safety critical application etc.). For instance the problem of the Byzantine fault (agreement protocol between three nodes) is well known from far back.

Hence there is no problem to design such agreement procedure. The difficulty comes when the testing procedure is necessary. Additional complexity is contributed by the requirement for fault tolerant agreement procedure. This means that occured fault in part of the system doesn't cause wrong reaction. The system has to mask the faults (if possible) and when the masking redundancy is exhausted a halt of the system is required. The reason for wrong reaction can be a fault which is unconsidered by the system design, i.e. the system has a design error.

The proposed in the paper test algorythm provides a tool for checking of all possible combinations between design errors and system faults. The system is divided in some sub nodes and one or more of them are considered as faulty. Followed by system test intended for detection of a wrong reaction. For this purpose the system is modeled with attributed Petri net.

It generates all correct and erroneous messages (called injected messages). An operational fault is characterized by the set of messages injected during a single test run. After these messages have been sent to the faultfree rest of the distributed system under test, it is checked whether they are covered by the fault processing technique or lead to illegal output possibly causing a wrong reaction. Injection is carried out completely independent of the underlying local operational fault in the sender, because the reaction of the faultfree system part is to be tested, not the internal behavior of a faulty sub node.

In contrast to other injectors used for quantification of dependability measures [1, 6] the injectors for testing need not be realistic.

[1] Taschko Nikolov is with Telecom Department at the Technical University of Sofia, "Kliment Ohridsky Blvd 8, 1756 Sofia, Bulgaria, e-mail: tan@vmei.acad.bg

II. ATTRIBUTED PETRI NET

The attributed Petri net model comprises places with capacity 1 and transitions: firing of transition changes the marking, i.e. the presence of tokens in places, to express a sequence of states. Only the decision of the agreement procedure on a single result is modeled. However, the model covers all possible combinations of data, time and fault scenarios. The model is finite with respect to the token game. It is infinite because of the following time and data attributes:

Time attributes model message delays and timeouts

Time variables t_x are associated with places to express that the token is "frozen" there for duration t_x and released afterwards. No concrete value to t_x or a probability distribution is assigned. Instead it is specified an interval $[a_x, b_x]$ with constants a_x and b_x to mark the lower and upper bounds for t_x, i.e. the range where t_x can vary non-deterministically. Fig. 1 shows the time variables t_M and t_T for message ***M*** and timeout ***T***, respectively.

Data attributes model message contains local variables. Data items are represented by variables, not by concrete values – see d_A, d_M and d_B in fig. 1. After firing functions may be applied to data expressions and assigned to data variables, see $d_M = f(d_A)$, $d_B = g(d_M, d_B)$ and $d_B = 0$ in fig. 1. Predicates and functions are treated symbolically as well.

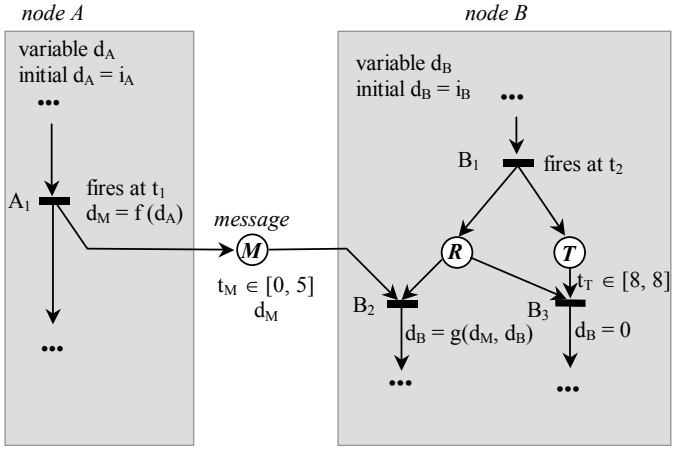

Fig. 1. Part of an attributed Petri net model

Nodes as a whole form operational fault region and, on fault occurrence, exhibit wrong behavior. Unlike most other Petri net fault models [4], this model does not add fault elements like "message loss transitions". Instead, when assuming node X as faulty it will be simply cut the respective part from the model, including the Petri net edges leading into and out of the faulty node. The places representing messages sent by the faulty node stand for *injected messages*. Each such place IM is marked initially and attributed with a time

variable $t_{IM} \in [0, \infty)$ and condition-free data variables to cover any wrong behavior.

Cutting a substantial part from the attributed Petri net does not only provide a general model of arbitrary operational faults. It also makes the analysis more efficient.

The modeler has to define correctness by use of *correctness predicate* over sequences of marking.

The formal framework for specification of the *correctness predicate* is defined as follow:

- Let $P = \{p_1, \ldots, p_{|P|}\}$ be the set of $|P|$ *places* in the attributed Petri net.
- Let m be a marking, $m \subseteq P$, where $p_i \in m$ means that p_i carries a token in m.
- Let $rM = \{m_1, \ldots, m_{|rM|}\}$ be the set of $|rM|$ *reachable markings*, with $m_i \subseteq P$
- Let $Q = \{q_1, \ldots, q_{|Q|}\}$ be the set of $|Q|$ reachable sequences of markings, which are subsequently reached according to the attributed Petri net. The initial marking is the same for all paths, e.g. $q_1(1) = q_2(1) = \ldots = q_{|Q|}(1)$.

It is obvious that the marking is only reachable if it is in any of the paths. Consequently, rM is the union:

$$rM = \bigcup_{1 \le i \le |Q|, 1 \le j \le r_i} q_i(j) \qquad (1)$$

The marking m can also be expressed by a Boolean vector, where element i is true if $p_i \in m$. The correspondence between the place set and the vector representation is expressed by the function: $v: rM \rightarrow \{true, false\}^{|P|}$

For the example from fig. 1 is obtained:

$P = \{p_1, p_2, p_3\}$, p1 = *M*, p2 = *R*, p3 = *T*,

$Q = \{q_1, q_2, q_3\}$ with path lengths $r_1 = r_2 = r_3 = 3$, see fig. 2.

First path:

$q_1(1) = \{M\}$, $q_1(2) = \{M, R, T\}$, $q_1(3) = \{T\}$.

Second path:

$q_2(1) = \{M\}$, $q_2(2) = \{M, R, T\}$, $q_2(3) = \{M\}$.

Third path:

$q_3(1) = \{M\}$, $q_3(2) = \{M, R, T\}$, $q_3(3) = \{M\}$.

The second and the third path only deviate in their attributes (not visible here). Reachable markings are: $rM = \{\{M\}, \{M, R, T\}, \{T\}\}$. The marking $m = \{T\} \in rM$, for example, can also be represented by the Boolean vector $v(\{T\}) = (false, false, true)$, because $p_3 = T$ assigning index 3 to *T*.

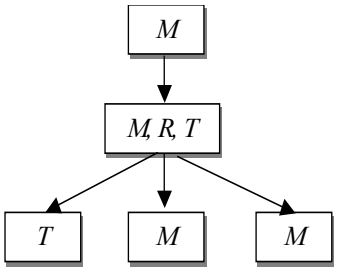

Fig. 2. Reachable paths for the model from fig. 1

III. System Model

A proposal of agreement procedure between two operational nodes is given (fig. 3). To ease the description and later fault injection the system is separated in sub nodes – Input (I), Node A (A), Node B (B), Comparator (C) and Switch (Output Node). From one side Node A and the Switch are physically one device (computer) and from another side Node B and the Comparator are combined.

The Input node has to transmit information for diverse actions to the Node A and the Comparator. Node A calculates the necessary action of the system, makes an absolute test of the data and tries to send them through the Switch to the environment. The comparator controls the Switch position and only the closed Switch is able to send the information.

Node B receives the calculated data from Node A and tries, according to the environmental status and the proposal from Node A, to "guess" what has been "desired" by the Node Input, e.g. to solve the inverse function (f^{-1}) of Node A.

The Comparator compares the result from Node B and the Input and if the comparison is true the Switch will be closed. Otherwise the Switch remains open.

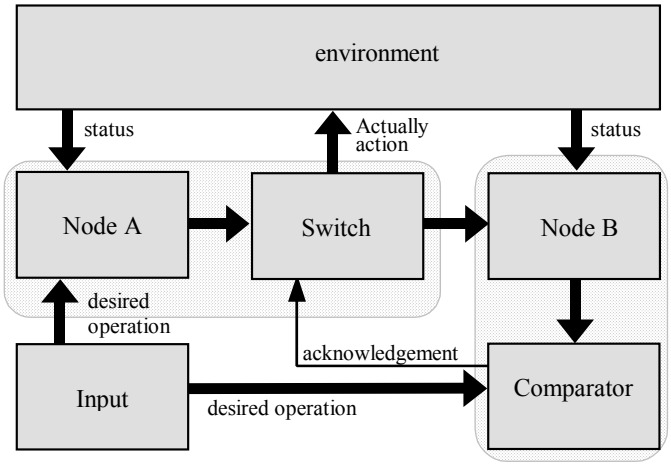

Fig. 3. Agreement procedure between two nodes

The Petri net model of the system is shown on fig. 4.

As it was mentioned above, a subject of the model is only the agreement procedure. The Input node can send either a sense or nonsense command. The sense command has to cause an active reaction; while the nonsense command has to cause halt. The simultaneous marking of sense or nonsense places means a nondesired (i.e. dangerous) reaction. Because of the Node A has an absolute test there for the token can mark only one of the output places (A_reaction or A_halt). The aim of Node B is only to calculate the inverse function. The comparator has to perform the following tasks:

- To check if nonsense command from the Input Node is coming and if it is true the expected command from the Node B should be halt_B;
- The condition to send an acknowledgement to the Switch is the absence of halt_B **and** a positive result from the comparison of sense_C and halt_B.

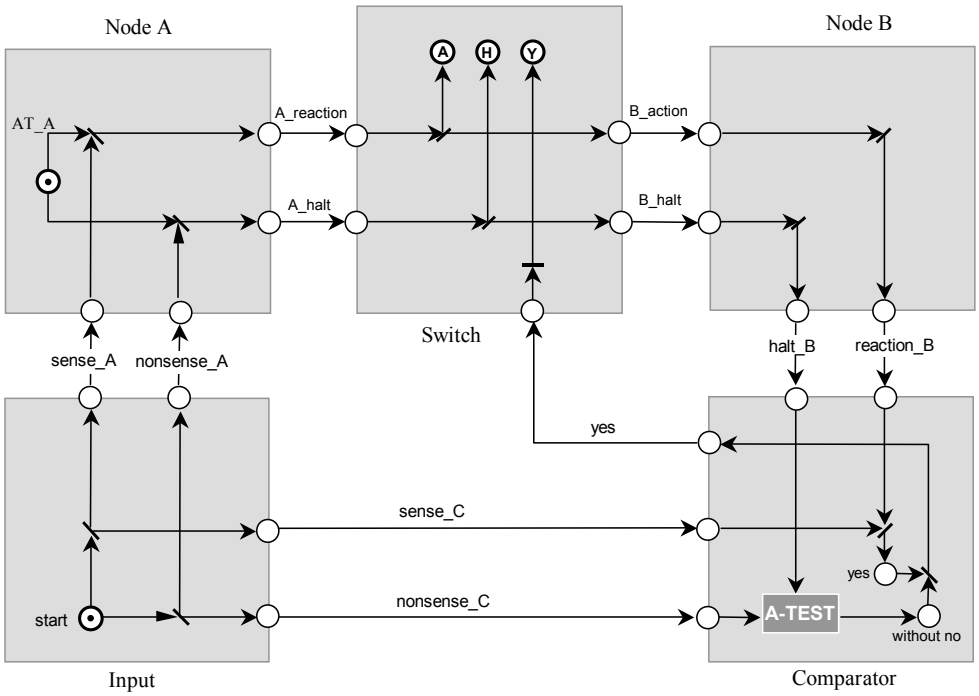

Fig. 6. Petri net model of Agreement procedure between two nodes

The marking of the output place „Y“ of the Switch confirms that the output reaction (action or halt) is correct. Presence of tokens in all output places (A, H, Y) can be regarded as a nondesired (dangerous) reaction.

IV. Model-Based Evaluation

Existing methods of fault/error injection into models use simulation models [5]. We developed a fundamentally different approach by staying in the model world of attributed Petri nets using respective analysis methods. Instead of simulating particular executions under injection, we check directly whether the fault time and data satisfy any of the „dangerous paths“ caused by a design fault. Fig. 5 shows this approach.

In the left column the process of generating the faults to be injected is shown from top to bottom: Firstly, reachability analysis with symbolic treatment of time and data attributes generates the path set Q. Then, one of the selection criteria (see next section) can be applied to form the path set Q_{sel}, where *sel* stands for *random*, *close-to-danger*, *coverage* or *none*. Finally solutions to the time and data conditions in the paths Q_{sel} are determined to form the fault set *faults*(Q_{sel}). It consists of k faults, each for exactly one of the k test runs.

In the right column a similar process is shown for an attributed Petri net model containing a design error – supposed to be revealed by fault injection. Reachability analysis leads to a path set Q'. From this set the paths, which violate the application-depend correctness specification are selected.

The set of such paths is called Q'_{danger} where $Q'_{danger} \subseteq Q$. If dangerous paths do not exist, e.g. $Q'_{danger} = \varnothing$, the evaluation stops here. Otherwise from Q'_{danger} the time and data conditions are extracted - *conditions* (Q'_{danger}). These time inequalities and data predicates in first order predicate logic express the conditions under which a dangerous state is reached. These formulae characterize the injected messages (points in time and data of injected messages).

Finally we have to determine the effect of injecting of a fault from *faults*(Q_{sel}) into thesystem containing a design error. More precisely: We have to decide whether the injections lead to a dangerous state, as expressed by *conditions* (Q'_{danger}). If not, the design error remains undetected. Otherwise the injection is considered as successful.

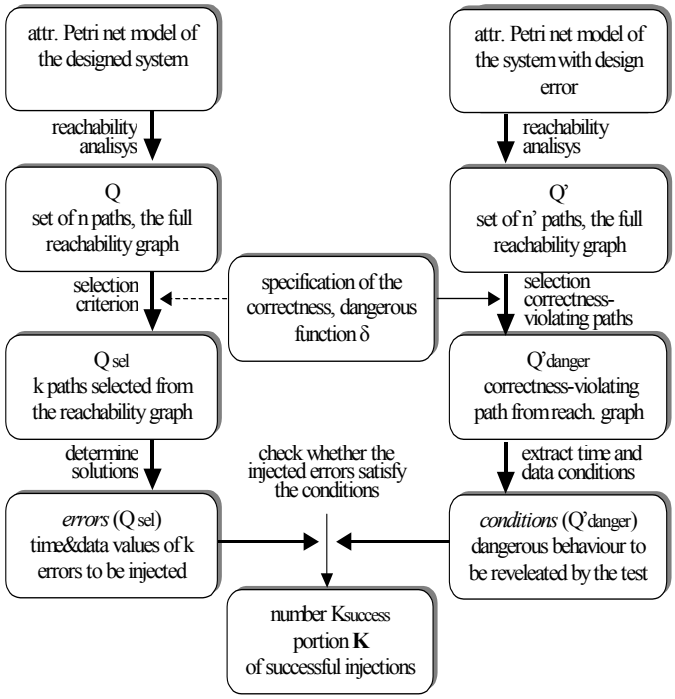

Fig. 5. Model-based evaluation of fault injection for testing

We denote the absolute number of successful injections by $k_{success}$ and the relative portion $k_{success}/k$ by **K**.

Basically, a single successful injection would be sufficient to reveal a design error. However, the coverage of the checks in the real system is never perfect.

IV. Fault Selection Criteria

Random selection: k out of n paths are selected according to the discrete uniform distribution. This criterion is only used for comparison

Close-to-dager: Paths are the more preferred as test cases the closer they come to danger. The danger conditions have to be defined in advance.

Coverage Selection: According to the known principles of white-box software testing [2, 3] one can try to execute all parts of the given test object. However, this strategy must be modified for Petri nets due to their non-sequential nature. For this reason an *activity* of a place in the Petri net is defined: Low activity, called *activity 0* means that the marking of a place remains unchanged throughout a path. *Activity 1* means a single change and all more frequent changes fall into the category *activity 2*. Paths with different place *activities* have to be selected.

V. Experiments and Results

Into the system have been inserted *design errors* and have been injected *operational faults*. According the method presented on fig. 5 the portion **K** of successful injections is obtained.

We generated four different *artificial design errors* for the modeled system:

a) Incomplete input checks in Node A, accepting wrong input data
b) Different design errors in the input checks of Node A and the Comparator
c) Same design error in the input checks of Node A and the Comparator
d) Wrong implementation of f^{-1} generates an arbitrary output if f(i) = „halt“

Single and double operational faults have been injected into the following sets of components: {I}, {A}, {B}, {C}, {I, A}, {I, B}, {I, C}, {B, C}. The path numbers of the model areas follows:

- $|Q| = 7$ in the absence of both design faults and operational faults,
- $|Q| \leq 1116$ in the presence of operational fault only, (none of them violates correctness: $|Q'_{danger}| = 0$
- $|Q| \leq 4250$ and $|Q'_{danger}| \leq 2946$ in the presence of both design errors and operational faults.

Fig. 6 shows the portion $\mathbf{K} = k_{success}/k$ of successful injections depending on the selected portion $\gamma = k/n$ of paths for the three selection criteria. Note: $\gamma = 100\%$ means that selection criteria are not applied.

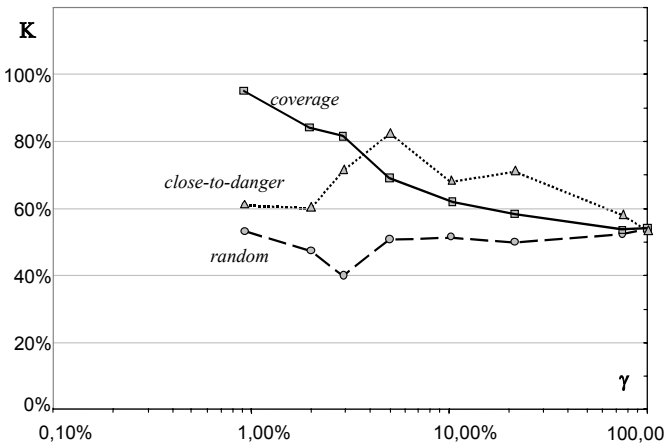

Fig. 6. Portion **K** of *successful injections* for the three selection criteria

VI. Conclusion

Specialized mechanisms are applied to attack the main problem of testing: to select appropriate faults among the members of the extremely huge set off all possible faults.

According to this principle the main design decisions are:

- Definition of the attributed Petri net model
- The main countermeasure against state explosion is the symbolic treatment of time and data attributes by linear inequalities and first-order predicate logic, respectively.
- Reduction in the number of faults is achieved by special selection criteria: Close-to-danger is directly based on the given definition for undesired behavior. Coverage uses the novel notion of place activity in the attributed Petri nets.

Experiments showed that the coverage criteria should be preferred. Since its definition is application-transparent, it can be applied to the test of several algorithms.

References

[1] U. Aldenhoff: Theorem prover to decide data conditions in a Petri net model of fault-tolerant distributed systems; Dipl. Thesis, (in German) University of Dortmund, 1996

[2] Y. Chen, K.Echtle, W. Görke: Testing Fault-Tolerant Protocols by Heuristic Fault Injection; Informatik-Fachberichte 283, Springer, 1991, pp. 407 - 418

[3] K. Echtle, Y. Chen: Evaluation of Deterministic Fault Injection for Fault-Tolerant Protocol testing, FTCS-21, Dig. of P., 1991, pp. 418 – 425

[4] C. Girault: Proof of Protocols on the Case of Failures; Parallel Processing systems, an Advanced Course, Cambridge U. Press, 1982, pp. 121 – 139

[5] E. Jenn, J. Arlat, M. Rimen, J. Ohlson, J. Karlsson: Fault Injection into VHDL Models: The MEFISTO Tool; FTCS-24, Digest of Papers, 1994, pp. 66 – 75

[6] A. Steininger, H. Schweizer: A Model for the Analysis of the Fault Injection Process; FTCS-25, Digest of Papers, 1995, pp. 186 – 195.