TEST GENERATION FOR THE DISTRIBUTED TEST ARCHITECTURE

(Extended abstract)

Gang Luo*, Rachida Dssouli*, Gregor v. Bochmann*,
Pallapa Venkataram** and Abderrazak Ghedamsi*

* Departement d'IRO, Universite de Montreal,
C.P. 6128,Succ.A, Montreal, P.Q., H3C 3J7, Canada
e-mail:luo@iro.umontreal.ca, Fax: (514) 343-5834

** Dept. of Electrical Communication Engineering,
Indian Institute of Science, Bangalore-560 012, Indian.

1. Introduction and the distributed test architecture

Protocol testing is an important phase for ensuring the quality of communication networks; especially, testing distributed systems is an interesting issue in protocol engineering [14, 1, 18].

ISO (International Standardization Organization) developed the ISO distributed test architecture for testing layered protocols [10] (see Figure 1). Furthermore, a general distributed test architecture where the IUT (implementation under test) contains several distributed ports has been studied in [14]; it is used for testing distributed systems, based on the Open Distributing Processing (ODP) Basic Reference Model (BRM) (see Figure 2). In this architecture, the IUT contains several ports (i.e., points of control and observation), the testers cannot communicate or synchronize with one another unless they communicate through the IUT; and no global clock is available in the system. This architecture could model a test architecture of a communication network with n accessing nodes, where the testers reside in these nodes. When n=2, this general distributed test architecture reduces to the ISO distributed test architecture. We develop in the paper a test selection method with respect to this general distributed test architecture.

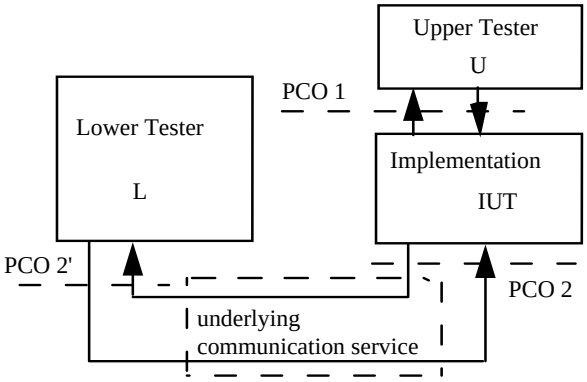


Figure 2.2. Distributed test architecture

Figure 1. ISO distributed test architecture

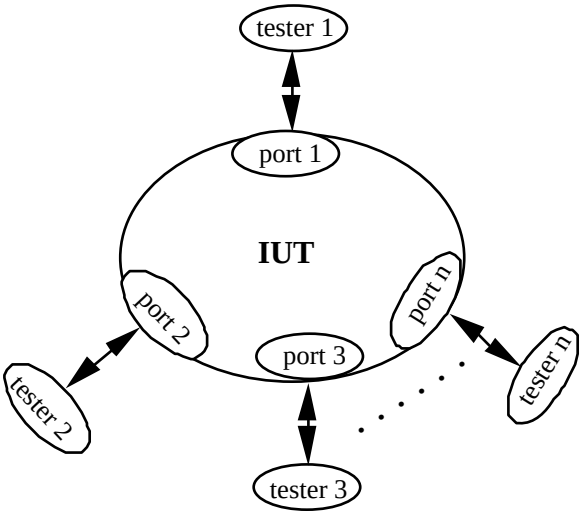


Figure 2. A general distributed test architecture

Usually, in the so-called local test architecture developed by ISO, the specifications of communication protocols are first abstracted into state machines [12], then test cases are generated from the resulting machines. A number of methods have been developed to generate test sequences for finite state machines (FSMs) [6, 16, 5, 17, 15]. However, they are not directly applicable to the distributed test architecture, because of *the synchronization problem* (as defined in Section 2) between distributed testers.

In the distributed test architecture, testing is relatively difficult because certain problems of synchronization between the testers may arise during the application of test sequences [18]. To solve this problem, an approach of test generation has been developed in [18] by modifying the existing test generation methods for FSMs such as the transition tour [15], the DS-method [11], and the W-method [5], using a concept of so-called synchronizable test sequences. Later on, further work has been done to either study the computation complexity of generating synchronizable test sequences [4, 2], or to propose some modification to that concept [9, 4].

However, these methods are all for the ISO distributed test architecture where there are only two ports, not for the general distributed test architecture where nε2. Furthermore, none of these studies addressed the issue of the fault coverage provided by their methods.

We present in Section 2 an approach to generating test sequences for FSMs with n distributed ports (np-FSMs) where nε2, which is a generalized version of the approach given in [18]. We also explore in Section 3 the issue of fault coverage in the distributed test architecture. In Section 4 we discuss the application of our method to generating test sequences for a so-called quorum-based protocol.

2. An outline of a test generation method

Under the architecture given in Figure 2, the testers are distributed over several sites, and they are only synchronized through the interactions with the implementation. In this situation, considering two consecutive transitions t1 and t2 of a given np-FSM I (nε2), one of the testers is said to face a *synchronization problem* if this tester did not take part in the first transition and if the second transition requires that it sends a message to I.

We now generalize the concept of synchronizable test sequences given in [18] to the case of np-FSMs, nε2, handling the synchronization problem of general np-FSMs. Given a np-FSM I with the ports 1, 2, ..., n, we require the following concepts for the ease of presentation.

DEFINITION *Interaction ports* (IP) *of a given transition* [pi, PO]**:**
Let pi□{1, 2, ..., n} and PO⁄{1,2, ..., n} (n is the number of ports). [pi, PO] is said to be an *interaction port* (IP) *of a given transition* t if t receives an input from the port pi and sends to each port in PO an output (if PO=□, t does not send any output). □

DEFINITION *Synchronizable test sequences***:** Given a pair of transitions t1 and t2 of I, let [pi, PO] and [pi', PO'] be their IPs, respectively. t1 and t2 are said to be *synchronizable* if (1) pi=pi', or (2) pi'□PO. A given test sequence is said to be *synchronizable* if any two subsequent transitions of the sequence are synchronizable. □

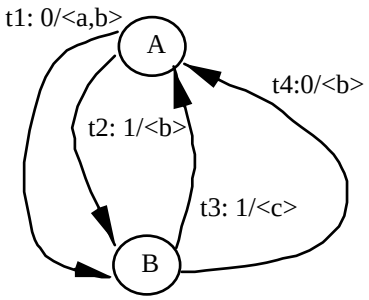
Guided by the idea of synchronizable test sequences, we modify existing test generation methods for FSMs such that the resulting test sequences are all synchronizable, using an approach similar to the one given in [18]. For instance, we give in the following a modified transition tour method.

Any graph traversal algorithm such as the one given in [19] can be modified to obtain a transition tour. Assume that an algorithm *TT* produces a transition tour; we present in the following a procedure for generating synchronizable transition tours, by modifying the algorithm *TT*.

Generating synchronizable transition tour:

Each transition *t* to be added to the sequence by the algorithm *TT* is first checked whether it forms a synchronizable pair together with the last transition *tt* of the sequence; that is, assuming the IPs of the transitions *tt* and *t* to be [pi, PO] and [pi', PO'], respectively, check whether pi=pi' or (2) pi'□PO are true. If the transitions *tt* and *t* are not synchronizable, a different transition from the present state is considered. If no suitable transition exists from the present state, the algorithm *TT* backtracks to the previous state, continuing the tour from there in a different way. This process continues until all the transitions of the machine are covered.

For example, Figure 3b shows a synchronizable transition tour for the 2p-FSM shown in Figure 3a. (Note: In Figure 3, Li and Lo represent the input and output alphabets, respectively)

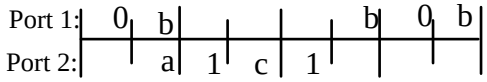


Li_1 = { 0 }, Li_2 = {1 }
Lo_1 = { b }, Lo_2 = { a}

(a)

Input sequence: 0, 1, 1, 0

Resulting global sequence:



IPs: <1,{1,2}>, <2,{2}>,<2,{1}>,<1,{1}>

tester1 -- port 1
tester2 -- port 2
The initial state is A

(b)

Figure 3: (a) An example of a 2p-FSM, (b) test sequence.

3. Fault coverage

We investigate in this section the issue of fault coverage in the distributed test architecture. We come out with a fact that the modified methods in the literature cannot ensure the same fault coverage as the corresponding original testing methods. For instance the modified transition tour cannot detect all output faults for FSMs of two distributed ports although a transition tour can detect all output faults for the FSMs with only one port. A class of output faults may remain undetected if the modified transition tour is applied to an FSM with two ports. Unfortunately, none of the former articles on generating synchronizable test sequences [18, 9, 4, 2] had realized such weakened fault coverage.

A. *A class of output faults undetectable by synchronizable test sequences:*

For FSMs, as we know, the transition tour [15] can detect all output faults if no transfer faults occur. However, for 2p-FSMs, a transition tour which is even a synchronizable test sequence does not necessarily detect all output faults. For example, the 2p-FSM shown in Figure 4(a) is a faulty implementation of the 2p-FSM shown in Figure 3(a); and it has only output faults. The input sequence "0,1,1,0" is a synchronizable transition tour, as described in Figure 3(b). When the input sequence "0,1,1,0" is applied to the faulty implementation, we obtain the global sequence (trace) shown in Figure 4(b). Although the global sequence for the original 2p-FSM is different from the global sequence for the faulty one, the difference cannot be seen at the two local ports (Ports 1 and 2) since no global clock is assumed in the distributed test architecture. Therefore, a modified transition tour does not necessarily detect all output faults for 2p-FSMs.

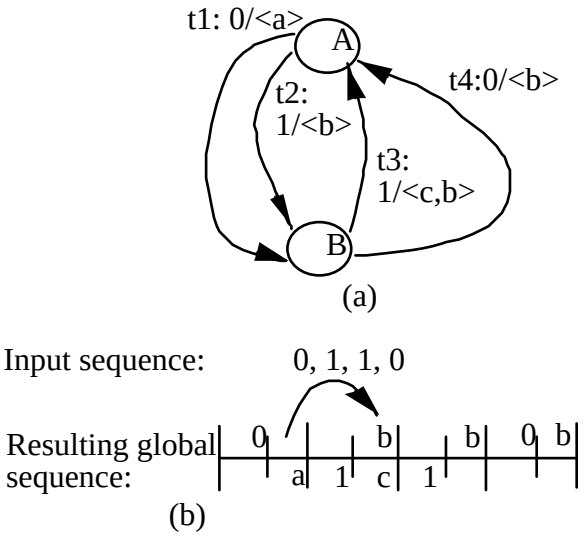


Figure 4: (a) An example of an output-shifting fault in a 2p-FSM, (b) the global, and local traces.

By using similar arguments, it is easy to prove that the modified W-method and DS-method [18] do not necessarily detect all output and transfer faults for 2p-FSMs.

We formalize the class of output faults which may remain undetected by the modified methods as follows: For two consecutive transitions, say t1 and t2, in a given specification S, the faulty implementation I can be obtained from S by removing an output from one of the two transitions and adding the output to the other transition. We say that this class of faults are *output-shifting faults*. The machine shown in Figure 4(a) has an output-shift fault with respect to the machine shown in Figure 3(a).

We present in the following an outline of improved approach which can detect the class of output faults:

1: Generate a set of synchronizable test sequences, say □, by using the approach given in Section 3, with respect to one of the test generation methods for FSMs such as the transition tour [15], the W-method [5], and so on.

2: Find a set of all transition pairs, say Ω, along the paths caused by applying the sequences of □, each pair of which may have an output-shifting fault which is undetectable using □.

3: If Ω is empty, stop. Otherwise, add a set of additional synchronizable test sequences to □, or concatenate some synchronizable sub-sequences to the sequences in □, such that □ can ensure the absence of output-shifting faults in the transition pairs of Ω. Go to Step 2.

Further detail can be found in [13].

4. Application of the test generation method to a quorum-based protocol

To illustrate the method we have taken a quorum-based protocol which is used for nonblocking of resources and increasing the distributed system's availability [8, 3]. We generate test sequences for this protocol. Figure 5 gives the architecture of a distributed system that uses a quorum-based protocol. The test architecture for testing this protocol is shown in Figure 6.

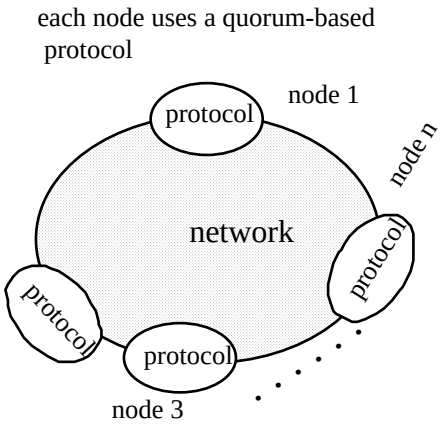


Figure 5. The architecture of the distributed system using quorum-based protocol

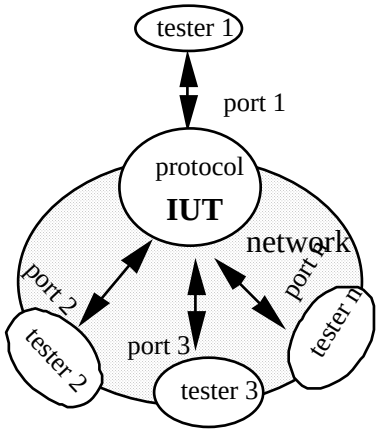


Figure 6. The test architecture for testing a quorum-based protocol

The basic idea of the protocol is that a majority of nodes in the distributed system must agree on the commit or abort of a transaction before the transaction is committed or aborted. The weights which are assigned to the ports are usually called votes, since they are used when a port "votes" on the commit or abort of a transaction. The basic rules of a quorum-based protocol are:

1. Each port, i, has associated with it a number of votes, v_i , and v_i is a positive integer.
2. Let V indicate the sum of the votes of all ports of the distributed system.
3. A transaction must collect a commit quorum Vc before committing.
4. A transaction must collect an abort quorum Va before aborting.
5. Va + Vc >V

In practice, one usually assumes Va + Vc =V+1.

We have modeled the IUT as a finite state machine and generated test sequences using the proposed method, as described in [13].

5. Conclusion

Test generation in the context of the distributed test architecture is an interesting issue in the area of quality assurance of the communication networks. A general distributed test architecture, which is a generalized version of ISO distributed architecture, has been presented in the literature for testing distributed systems based on the Open Distributing Processing (ODP) Basic Reference Model (BRM). Based on this test architecture, we develop a test generation method for systems modeled by finite state machines. We have applied this method to generate test sequences for a so-called quorum protocol.

Furthermore, we have investigated the issue of fault coverage related to test generation in the distributed test architecture. We point out a fact that the methods given in the literature for the distributed test architecture, modified from certain existing methods, cannot ensure the same fault coverage as the corresponding original testing methods. The distributed test architecture also gives rise to new problems of FSM based diagnosis, and the diagnosis method of FSMs of one port [7] may be modified for np-FSMs by applying the concept of synchronizable test sequences.

Acknowledgment: This work was supported by the IDACOM-NSERC-CWARC Industrial Research Chair on Communication Protocols at the University of Montreal (Canada).

REFERENCES:

[1] G. v. Bochmann, R. Dssouli and J. Zhao, "Trace Analysis for Conformance and Arbitration Testing",

IEEE Transactions on Software Engineering, Vol. SE-15, No.11, 1989, pp.1347-1356.

[2] Sylvia Boyd and Hasan Ural, "The Synchronization Problem in Protocol Testing and Its Complexity", Information Processing Letters Vol.40, No. 8, (Nov. 1991) pp.131-136.

[3] S. Ceri and G. Pelagatti, Distributed Batabases: Principles and systems, McGraw-Hill, New York 1984.

[4] Wen-Huei Chen, Ching-sung Lu, Jinn-Tuu Wang, and Richard-Jinjr Lee, "Constrained Chinese Postman Problem with Its Application on Synchronizable Protocol Test Generation", Journal of Information and Engineering Vol.6, (1990), pp.149-157.

[5] T.S.Chow, "Testing Software Design Modeled by Finite-State Machines, IEEE Trans. on Software Eng., Vol. SE-4, No.3, 1978.

[6] S.Fujiwara, Gregor von Bochmann, F.Khendek, M.Amalou & A.Ghedamsi, "Test Selection Based on Finite State Models", IEEE Trans. on Software Engineering, Vol SE-17, No.6, June, 1991, pp.591-603.

[7] A.Ghedamsi and G.v. Bochmann, "Diagnostic Tests for Finite State Machines", Distributed Computing Systems Conf. 92 in Japan.

[8] Maurice Herlihy, "A Quotum-Concensus Replication Method for Abstract Data Types", ACM Transactions on Computer Systems, Vol.4, No.1, Feb 1986, pp.32-53.

[9] Darrell Hubbard, "Deterministic Execution Testing of FSM-Based Protocols", AT&T Technical Journal, Vol.69, No.1, 1990, pp.119-128.

[10] ISO/TC97/SC21, OSI conformance Testing Methodology and Framework - Parts 1- 5, ISO, 1991.

[11] Z. Kohavi, Switching and Finite Automata Theory, New York: McGraw-hil, 1978.

[12] D.Y. Lee and J.Y. Lee, "A Well-Defined Estelle Specification for the Automatic Test Generation", IEEE Transactions on Computers, Vol.40, No.4, April, 1991, pp.526-542.

[13] Gang Luo, Rachida Dssouli, Gregor v. Bochmann, Pallapa Venkataram and Abderrazak Ghedamsi, "Generating Synchronizable Test Sequences Based on Finite State Machines with Distributed Ports", submitted for publication.

[14] J. de Meer, V.Heymer, J. Burmeister, R. Hirr and A.Rennoch, "Distributed Testing", Participants Proceedings of International Workshop on Protocol Testing Systems, Oct. 15-17th, 1991, the Netherlands, pp.IV43--51.

[15] S. Naito and M. Tsunoyama, "Fault Detection for Sequential Machines by Transition Tours", in Proc. IEEE Fault Tolerant Comput. Conf., 1981.

[16] Alexandre Petrenko and Nina Yevtushenko, "Test Suite Generation for a FSM with a Given Type of Implementation Errors", IFIP 12th International Symposium on Protocol Specification, Testing, and Verification, U.S.A.,North-Holland, 1992, pp229-243.

[17] K.Sabnani & A.T.Dahbura, "A Protocol Test Generation Procedure", Computer Networks and ISDN, Vol.15, No.4, 1988, North-Holland, pp.285-297.

[18] Behcet Sarikaya and Gregor v. Bochmann, "Synchronization and Specification issues in Protocol Testing", IEEE Transactions on Communications, Vol.COM-32, No.4, April 1984, pp.389-395.

[19] R.Tarjan, " Depth-first search and linear graph algorithms.", SIAM J. Comput., vol.1, no.2, 1972.