11

Protocol Conformance Testing – A Survey

B. Neelakantan and S.V.Raghavan,
Department of Computer Science and Engineering,
Indian Institute of Technology Madras, Madras 600 036, India.
svr@iitm.ernet.in

Abstract

Testing is an integral part of protocol development cycle. In this paper, we will briefly discuss the protocol conformance testing methodologies and framework proposed by the International Standards Organization. Many efficient test sequence generation methods have been proposed to check the conformance of an implementation of a protocol to its standards. We will discuss these methods briefly. Finally, we will compare different test methodologies based on their fault coverage and the length of the test sequence.

Key words

Black Box Testing, Fault Detection Methods, Fault Diagnosis Methods, Finite State Machine, Protocol Conformance Testing, Test Sequence, Test Suite Design.

1 INTRODUCTION

Protocol Software Engineering has been an active area of research in the field of computer networks and distributed computing. From the time International Standards Organization (ISO) started developing standards for conformance testing of network protocols, a lot of research work has been undertaken in the field of conformance testing.

S. V. Raghavan et al. (eds.), *Computer Networks, Architecture and Applications*

Conformance testing involves testing both the capabilities and the behavior of an implementation and checking what is observed against the conformance requirements in the relevant recommendations and against what the implementor states the implementations capabilities are [Rayner (1987)]. This is done by applying a test suite to the Implementation Under Test (IUT) and giving a verdict by comparing the observed output with the expected output. Conformance testing *does not* asses the performance, robustness or reliability of an IUT.

The protocol specification and its implementation are modeled as deterministic Finite State Machines (FSMs). Also, the implementation of a protocols is considered a *black box* for testing purposes. The internal states of the FSM are not visible and they can be identified only by applying the State Identification Sequences (SISs) or signatures for that state. The aim of the testing process is to find out whether all the states that are in the specification are also there in the IUT and if all the transitions are implemented correctly. This is equivalent to *checking experiments* of hardware testing.

In this paper, we present two important aspects of protocol conformance testing. The first part given in section 3, discusses the Open System Interconnection (OSI) conformance testing standards, the abstract test suites, the test architecture, the test language, test realization and conformance assessment review. These standards have approached the International Standards stage. The second part discusses the test sequence generation for the protocols. This part is yet to be standardized and there is a lot of research work going on in this area and are discussed from section 4 to section 7.

The paper is organized as follows. In section 2, we give some definitions that are used in the paper. In section 3, we discuss the OSI conformance testing methodology and framework. In section 4, we highlight the issues that should be considered during the test sequence generation. In section 5, we give different testing methodologies, which are capable of detecting a faulty IUT. In section 6, we discuss fault diagnostic methods. In section 7, different testing methods are compared with different factors like fault coverage and length. Finally, in section 8, we summarize the work reported in this paper.

2 PRELIMINARIES

2.1 Finite state machine and graph theory

A DFSM M is a quintuple $M = (I, O, S, \delta, \lambda)$, where I, O, S are finite, nonempty sets of inputs, outputs and states respectively; $\delta : I \times S \longrightarrow S$ is the state transition function; λ is the output function such that $\lambda : I \times S \longrightarrow O$. The cartesian product $I \times S$ is the set containing all pairs of elements (I_i, S_j). The state transition function δ associates with each pair (I_i, S_j) an element S_k from S, called the *next state*. The output function λ associates with each pair of (I_i, S_j) an element O_k from O [Kohavi (1978)]. Henceforth in this paper we assume that the FSM's are deterministic.

A *completely specified* FSM is one in which there exists a permissible output ($\in O$) for every input element ($\in I$) at every state. Otherwise, the FSM is said to be *partially specified.* If for every pair of states s_i and s_j of the FSM, there exists an input sequence which takes the FSM from s_i to s_j, the FSM is said to be *strongly connected.* An FSM is *minimal* if no two states of the FSM produce identical output for any possible input.

An FSM can be represented by a directed graph $G = (V, E)$, where the set of nodes $V = \{v_1, v_2, \ldots, v_n\}$ represent the set of specified states of the FSM and E, the set of labeled directed edges, defined as follows: There is a directed edge $e_{ij} \in E$, from v_i to v_j ($v_i, v_j \in V$) labeled a_k/o_l, if and only if there is a transition from v_i to v_j with input label a_k and output label o_l. Since the FSM is deterministic, for each node v_i there are no two edges leaving v_i, with same input labels. We assume that the specification FSM have n states and m edges.

A *transfer subsequence* between any two states is the shortest path between those two states. An input sequence is said to be a *homing sequence*, if the state of the machine after application of the input can be uniquely determined from the machines response, regardless of the initial state [Kohavi (1978)]. A *recovery subsequence* takes the FSM from the home state to a state where from the next test subsequence starts.

A machine is k-distinguishable if, for each pair of states, there is at least one input sequence of length k which, when applied to the pair, yields different output sequences [Chow (1978)]. A faulty transition in an IUT is *directly reachable* by a test subsequence if no other fault is encountered till the input corresponding to *edge under test* is applied [Bochmann (1993)]. Some testing methods assume that a *reliable reset* is present in the IUT, which when applied to the IUT takes it to the *start state*, irrespective of the state in which the IUT exists.

2.2 The FSM fault model

The conformance testing methods consider the specification and the IUT as FSM's and centers around the idea of applying sequence of inputs to the FSM's. If the *observed output* of the IUT is the same as the *expected output*, the IUT is said to have passed the test. However, if the *observed output* differs from the *expected output*, a *fault* is said to have occurred. Faults in an IUT can be classified as follows [Bochmann (1991)].

Output fault : A transition has an output fault if, for the corresponding state and received input, the IUT provides an output different from the expected output.

Transfer fault : A transition has a transfer fault if, for the corresponding state and received input, the IUT enters a different state other than the one specified by the next-state function.

Missing transition fault : An implementation has an missing transition, if for a pair of present state and input, the transition is not present in the IUT.

3 CONFORMANCE TESTING FRAMEWORK

In this section, we will discuss the OSI conformance testing methodology and framework, which is a five part standard. The aim of this standard is to have uniformity in testing the OSI protocols. This will lead to comparability and wide acceptance of test results produced by different testers. A survey of developments in conformance evaluation methodology, test architecture, test language, test generation methodology, formal description techniques and other issues related to protocol testing can be found in [Linn (1989), Rayner (1987)].

3.1 Part 1 : General concepts

Part 1 of the standards specifies the applicable requirements that are to be met by any implementation. According to OSI, conformance is concerned only with the conformance of an IUT with respect to its standards. The conformance requirements are given as follows: (a) Mandatory requirements, which are to be met in all cases; (b) Conditional requirements, which are to be observed only when the conditions specified in standards apply; and (c) Options, which can be selected to suit the implementation [ISO (1989a), Rayner (1987)].

The client should provide two documents to the testing center which will help in obtaining the test suite for the IUT. Protocol Implementation Conformance Statement (PICS) is a statement provided by the implementor, stating the capabilities and options which are implemented, and features which are omitted. Protocol Implementation Extra Information for Testing (PIXIT) contains the information needed by the test operator in order to run the appropriate test suites. It should also provide other specific information which cannot be provided in the PICS. PICS and PIXIT should not contradict each other.

ISO distinguishes four types of testing according to the extent to which they provide an indication of conformance. They are the

1. *Basic interconnection test* checks whether there are any gross violations in the IUT.
2. *Capability test* verifies whether the observed capabilities of the IUT are same as mentioned in the PICS.
3. *Behaviour test* aims to provide a comprehensive testing over the full range of dynamic conformance requirements that are specified in the standard.
4. *Conformance resolution test* does in-depth testing of an IUT to particular requirements and provide a definite yes/no answer and diagnostic information to the specific conformance issues.

The *observed test outcome* is the output provided by the IUT for the applied *test case.* The *foreseen test outcome* is derived from the abstract test case along with the standards.

A *test verdict* is a statement of *pass, fail* or *inconclusive* and it is associated with each *test case.* A real system that is implemented based on OSI protocols, comes from a wide variety of configurations and hence a range of testing methods are defined by ISO, to take care of these variations. The IUT is a part of System Under Test (SUT) which is to be tested for conformance. ISO provides guidelines for testing single layer IUT's and multilayer IUT's [ISO (1989a)].

3.2 Part 2 : Abstract test suite specification

This part gives an approach to provide conformance test suites which are independent of the means of executing those test suites (called as Abstract Test Suites (ATS's)). As a result, the ATS can be useful in comparing the results produced by different organizations which run the corresponding executable test suites [ISO (1989b), Rayner (1987)]. An ATS comprises of a number of test cases. A test case shall include different types of tests like capability test, behaviour tests, etc. Each ATS should have a test purpose which tests a single conformance requirement.

Abstract test methods are described in terms of the Point of Control and Observation (PCO) available to both the Upper Tester (UT) and Lower Tester (LT). The location of PCO differs in different testing methods, and all the methods assume that there is at least one PCO for the LT. There are four types of abstract test methods that are recommended by ISO. They are the (i) Local test method, (ii) Distributed test method, (iii) Coordinated test method and (iv) Remote test method. A new test method was proposed by Zeng, *et al.*, called the Ferry clip method [Zeng (1989)]. All these test methods are explained in detail in [ISO (1989b), Rayner (1987), Zeng (1989)].

3.3 Part 3 : The tree and tabular combined notation

TTCN is an informal notation, through which generic and abstract test cases can be expressed. This notation is independent of the test methods, layers and protocols. TTCN test suites are hierarchically structured. A test suite can be viewed as a combination of test groups which in turn is a combination of test cases and this can be subdivided into test steps and test events [ISO (1989c)].

A TTCN suite has four parts: (i) the Test suite overview part, (ii) the Declarations part, (iii) the Dynamic part and (iv) the Constraint part. The test suite overview gives the purpose of test suite and its individual tests. Declarations are made about PCO, global variables, service primitives and PDUs. The dynamic behaviour table contains behaviour description, label, constraint reference, verdict and comment columns. The values for ASPs and PDUs that are used in the dynamic behaviour tables are given in the Constraints table [ISO (1989c)].

3.4 Part 4 : Test realization

Test realization is the process of producing a means of testing IUT's for conformance to standards with reference to an ATS. The means of testing is a combination of equipment and procedures that can perform: (a) the derivation (b) the selection (c) the parameterization and (d) the execution of the test cases, in conformance with the reference ATS and can produce a conformance log.

In the derivation process, abstract test cases are converted to executable test cases. In the selection process, the appropriate test cases for the IUT are selected based on PICS and PIXIT. In the parameterization process, parameters in the test cases are given appropriate values. Executable Test Suites (ETS's) are obtained after the derivation process [ISO (1989d)]. This part provides guidelines on the realization of Lower Tester, Upper Tester, Test Coordination Procedures, Conformance logs and other details.

3.5 Part 5 : Conformance assessment process

In this part, the role of both the client and the test laboratory are addressed. A broad set of guidelines are provided on the requirements that are to be met both by the client and the test laboratory [ISO (1989e)]. The conformance assessment process consists of three phases. They are:

1. *Preparation for testing :* In this phase, the general administrative steps are taken. Also, the completeness of the documents like PIXIT, PICS are verified.

2. *Test operations :* During this phase, the static conformance review is done. The parameterized executables are identified and they will be executed. The tests like basic interconnection tests, capability tests and behaviour tests are conducted.

3. *Test report production :* This phase is the culmination of the above two phases. Based on the results, an assessment is made about the conformance of the IUT with respect to the standard. This is recorded in System and Protocol Conformance Test Report (SCTR and PCTR).

4 TEST SEQUENCE GENERATION

The test sequence should be *efficient*, i.e., it should be relatively short, fast and easily executable. Also it should also be *effective*, i.e., it should have a fairly large fault coverage. These two properties are conflicting in nature.

The specification usually provides only the legal transitions for all inputs. These transitions are called *core edges*. A lot of state-input combinations are left unspecified and different implementations will have different behaviour for those state-input combinations. These unspecified state-input combinations are called *non-core edges* and it is assumed

that the IUT either goes to an error state or remains in the same state and generates null output. This is called the *completeness assumption* [Sabnani (1988)]. Two levels of conformance can be defined for partially specified FSMs. *Strong conformance* tests both core and non-core edges for conformance. *Weak conformance* tests only core edges for conformance.

Each state in the specification is identified by means of a SIS. There are three well known SISs: Distinguishing sequences (D), Characterizing set (W) and Unique Input/-Output sequences (U). An input sequence is a distinguishing sequence of the specification, if the output produced in response to the input sequence is different for each starting state in it [Gonenc (1970)]. The characterization set consists of a set of inputs that can distinguish between every pair of states present in the specification [Chow (1978)]. An Unique Input/Output sequence for a state is the input/output behaviour that is not exhibited by any other state of the specification [Sabnani (1988)]. Multiple SIS's may be present for a state. The test sequence methods use any one of the SIS to identify the states.

The aim of testing is to verify whether the IUT is *isomorphic* to the specification. The IUT is assumed to have same number of states as the specification. For each edge, a test subsequence is derived which should verify whether the *head* and *tail* state of the *edge under test* is the same as that of the specification. Also the *input/output behaviour* of the edge should be the same as that of the specification. Thus, the test subsequence has three parts: *a preamble, the edge under test* and *a postamble.* The *preamble* is an input sequence which will take the IUT to the *head state* of the *edge under test.* The *postamble* is the SIS (any one of D, W and U subsequences) for the *tail state* of the *edge under test.* Since the IUT is a black box, the *preamble* and *postamble* are usually long.

The specification should be *strongly connected, minimal* and in most of the cases the IUT is assumed to have *reliable reset.* Test sequence generation methods can be classified into *Fault detection methods* and *Fault diagnosis methods.* In fault detection methods, the aim is to simply find out if the implementation is faulty. Test result analysis does not give any indication of where the error is present. On the other hand, fault diagnosis methods will identify and locate the faulty transition.

5 FAULT DETECTION METHODS

5.1 Transition tour method

A *Transition tour* is an input sequence which takes the specification from its initial state, traverses every transition at least once, and returns to its initial state. If the FSM is strongly connected, a transition tour exists [Naito (1981)]. This method can detect output faults. On the other hand it may fail to detect transfer faults, as it does not verify the head and tail states of all the edges.

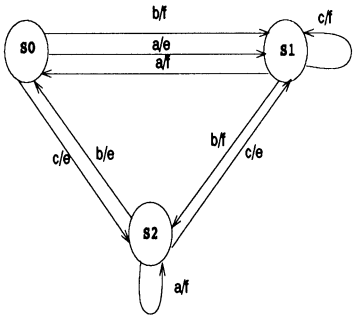

Figure 1: The specification graph.

5.2 Extended transition tour method

The aim of this method (E-method) is to verify if the behavior corresponding to each incoming edge and outgoing edge of every state in the implementation is similar to that of the specification [Kripanandan (1990)]. To achieve this, each incoming edge of the node is followed by every outgoing edge of the node, at least once in the test sequence. This method can detect output faults. It may fail to detect transfer faults if the specification is not 1-distinguishable.

5.3 Test sequence for D W and U methods

In the D, W and U methods, the testing is divided into two phases. Initially, the state verification is done by checking whether all the states that are in the specification are also there in the implementation. Next all the edges are tested by applying their corresponding test subsequences. The test subsequence for each transition e_{ij} is $P_{xi}.e_{ij}.SIS(v_j)$ where P_{xi} is the *preamble* to bring the graph to the *head state* of the *edge under test* from the *start state* and $SIS(v_j)$ is the SIS (D, W or U sequences) of the *tail state*. The complete test sequence is the set of test cases, which includes state verification subsequences and test subsequence for each edge in the specification. If any test subsequence is completely contained in another test subsequence, it can be eliminated.

Discussion and Example : Consider the specification given in Figure 1. Some of the DS's of the specification are {a.a}, {b.b} and {c.c}. The UIO sequences for the state *S0, S1 and S2* are a/e, $a/f.a/e$ and $a/f.a/f$ respectively.

Some of the W-sets for the specification are {a, a.a}, {b, b.b} and {c, c.c}. Let us use UIO sequences for testing. The state verification subsequences are given in Table 1 and the complete test sequences are given in the Table 2.

Table 1: State identification sequence verification

State	Test subsequence	Expected output
S0	r,**a**,**a**	null,**e**,**f**
S1	r,a,**a**,**a**	null,e,**f**,**e**
S2	r,c,**a**,**a**	null,e,**f**,**f**

Table 2: Test subsequence for all the edges

Edge	Test subsequence	Expected output	Edge	Test subsequence	Expected output
S0–a/e–S1	r,**a**,a,a	null,**e**,f,e	S0–b/e–S1	r,**b**,a,a	null,**f**,f,e
S0–c/e–S2	r,**c**,a,a	null,**e**,f,f	S1–a/f–S0	r,b,**a**,a	null,f,**f**,e
S1–b/f–S2	r,b,**b**,a,a	null,f,**f**,f,f	S1–c/f–S1	r,b,**c**,a,a	null,f,**f**,f,e
S2–a/f–S2	r,c,**a**,a,a	null,e,f,f,f	S2–b/e–S0	r,c,**b**,a	null,e,**e**,e
S2–c/e–S1	r,c,**c**,a,a	null,e,**e**,f,e			

5.4 Optimization of testing techniques

The complete test sequence for the IUT is obtained from the test subsequences. All the test subsequences generated in section 5.3 start from the start state. As a result, the test sequences have a lot of redundant transitions. Also there is no need to start the test subsequence from the start state. Instead of reset, a transfer subsequence can be used between different test subsequences. In general the problem of identification of an optimal test sequence from the test subsequences is *NP Complete* [Boyd (1991)].

An efficient algorithm exists if the IUT is assumed to have certain properties. By assuming that the IUT has a *reliable reset* or *self loop transitions* on all the states, Aho *et al.*, gave an efficient algorithm based on *UIO sequences* and *rural chinese postman algorithm* to determine the minimum length test sequence from the test subsequences [Aho (1988)].

Multiple SIS (MSIS) exists for some of the states of the IUT which ends at different tail states, and the length of the test sequence produced by using MSIS are in general shorter than the test sequence produced by using single SIS per state [Shen (1989)]. The reduction in the length of the test sequence will be more pronounced only when there are more number of SIS sequences per state which end at different tail states. An improvement to W method called W_p, was proposed in [Fujiwara (1991)]. In this method, instead of using the set W to verify the reached state, a subset of this set is used.

It was pointed out in [Aho (1988)] that, any overlap between different test subsequences can also be used to reduce the length of the test sequence. The papers [Miller (1993), Chen (1990), Yang (1990)] uses both the presence of MSIS and overlap between different test subsequences. The method given in [Miller (1993)] provides near-optimal test sequences in polynomial time.

Now we will briefly describe the algorithm given in [Miller (1993)]. A specification is said to be *nonconverging* if there are no two transitions entering the same state with the same input/output label. If the specification is nonconverging and has an *euler tour*, then the test sequence is any *euler tour* starting from the start state followed by $SIS(v_1)$, where v_1 is the start state. If the specification does not have an *euler tour* and it has an *euler path* starting and ending with the states v_k and v_l respectively, then the test sequence is any *euler path* of the specification from v_k to v_l followed by SIS_l. If the specification does not have an *euler path*, the specification is augmented with minimum number of edges to get an *euler path*. The test sequence is the *euler path* appended with the SIS sequence of the tail state of the path.

If there are converging edges in the specification, they are removed from the specification and edge-disjoint paths are computed from the resultant subgraph. For each edge-disjoint path, if the SIS sequence of the tail state of the path is appended to it, then all the edges that are in the disjoint path are verified. For all the converging edges, a test subsequence is generated by appending the SIS sequences of the tail state of the edges. These edge disjoint paths and the test subsequence corresponding to the converging edges are combined together in an optimal way by exploiting maximal overlap and minimal transfer subsequence. The detailed algorithm can be found in [Miller (1993)].

The lower bounds on the length of the test sequence based on D method *with* and *without overlap* between the test subsequences is given in [Ural (1993)]. An algorithm to obtain the test sequence using overlap between different test sequences are also given in [Ural (1993)].

In general, to obtain the shortest test sequence, *the length of the SIS should be minimum, multiple SISs should be used, the overlap between different test subsequences should be maximum and the transfer sequence between different test subsequences should be minimum.*

5.5 Regular checking method

The methods (D, W, U and E) combine the test subsequences without imposing any constraint on the ordering of the test subsequences. As a result, one fault may hide the presence of another fault and the faulty IUT may pass the test sequence. In this method (R-method) [Miller (1992)], a detailed study has been made on how the ordering of different test subsequences affects the fault detection capability. Different cases were

identified, where one fault hides another fault. Care was taken so that all such hiding faults were detected.

A brief outline of the R-method is given here. For each edge e_{ij}, the effect of transfer faults on all $(n-1)$ states is identified. A test subsequence is generated which will detect all such errors. This step is repeated for all the edges in the specification. Finally all the test subsequences are combined in an optimal way to obtain the test sequence. Combining different test subsequences is made as per ordering constraints imposed by the structure of the specification graph. A detailed analysis of how the ordering of test subsequences affects the fault coverage and an algorithm for generating the test sequence can be found in [Miller (1992)].

5.6 Analysis of fault coverage

The U method may fail to detect some transfer faults if the U sequences are no longer the U sequences in the IUT [Chan (1989)]. Hence it was suggested that all the U sequences should also be verified to be unique in the IUT in addition to the state verification phase. Further testing can be continued only if there is no error detected during this phase. This added overhead is acceptable, as it provides better fault coverage. It was shown in [Lombardi (1992)] that UIO verification need not be done for all states and they can be computed only for those states where such extra testing is necessary.

The test sequences generated based on the methods given in [Aho (1988), Shen (1989), Chen (1990), Yang (1990), Miller (1993)] may fail to detect some faults since, the UIO sequences of the specification may not be the UIO sequences for erroneous IUT. Hence it was suggested in [Yao (1993)] to include state verification subsequence too while generating the test sequence for the IUT. The length of the test sequence generated based on this method is generally longer than the U method, as certain portions of a test cases are repeated several times in order to make sure that the same state is reached in different points of the test case.

The α, β, γ, approximations to protocol testing were proposed in [Sidhu (1990)]. For each approximation there is a corresponding test subsequence. A test subsequence is defined most generally as $L_{ij}.CS(v_j)$, where L_{ij} is a sequence of input/output labels that take the specification from state v_i to state v_j and $CS(v_j)$ is the characterization subsequence. If length of L_{ij} is zero then it is called an α subsequence and if the length is one then it called a β subsequence and so on. It was suggested that for better fault coverage, higher order approximation should be used while generating the test sequence.

It was pointed out that the test sequence generated based on the same method and also having the same length, has different fault detection capabilities [Sidhu (1990)]. This is because there is considerable arbitrariness in obtaining the test sequence from the test subsequences, which affect the fault detection capability. To get better fault coverage,

the test sequence should have higher approximation in test subsequence. It was shown in [Miller (1992)] that only for some special classes of specifications one need to consider higher order of approximations, and in most of the cases, β subsequences will detect the faults. A survey of fault detection methods can be found in [Bosik (1991), Sarikaya (1982), Sidhu (1990), Sidhu (1989)].

6 FAULT DIAGNOSTIC METHODS

6.1 Fault resolution approach to protocol testing

This method (FR-method) assumes that for each transition e_{ij}, the label corresponding to e_{ij} should not occur in the SIS portion of the test sequence. This ensures that the erroneous e_{ij} does not mask the error in the SIS. Also, the IUT is assumed to have either output fault or transfer fault in at most one edge. This method assumes the presence of reliable reset in the IUT. The algorithm is given in [Ramalingam (1992)].

6.2 The diagnostic approach to protocol testing

In this method (DA-method), the initial testing is done by one of the following D, W, U methods. The basic assumption in this method is that all the faulty transitions are directly reachable by a test subsequence. The presence of *reliable reset* in the IUT is assumed. In this method, the observed outputs are compared with the expected ones. The difference, called *symptoms*, are identified for all the test subsequences. A set of edges where the fault might be present are identified based on the symptoms. Then, a set of *fault hypotheses* for each test subsequence is constructed. Fault hypothesis is the assignment of a specific fault to suspected edges. By changing the specification machine as suggested by fault hypothesis, if the observed output got for the original input sequence is the same, then the diagnostic candidate explains all the faults. This method may produce a number of diagnosis. Hence, further testing must be done in order to reduce the number of diagnosis. A detailed algorithm is given in [Bochmann (1993)].

6.3 Constraint satisfaction approach to protocol testing

This method generates a unique test sequence for the given specification which can distinguish any faulty IUT with the same number of states and the same number of inputs. The test sequence is incrementally generated from a set of test subsequences which represents the constraints imposed on the overall structure of the specification. In general, it is impossible to provide a unique test sequence for any specification [Kohavi (1978)]. But by making assumptions that, the specification is minimal, completely specified, strongly connected and the number of states in the IUT is equal to that of the specification (or

within some upper bound) it is possible to generate a test sequence which is unique to the specification.

The test sequence generated based on the CSP method has two parts. They are (i) TS_o, the initial subsequence which is a transition tour, i.e., it covers all the edges that are in the specification and (ii) TS_d, the test subsequence which distinguishes the specification from all other erroneous specification. The algorithm to generate the test sequence for the CSP method is given [Vuong (1990)].

6.4 Multilevel method

By applying only a subset of inputs to the IUT, a subgraph of the specification will be taking part in the test. M-method uses this concept for testing the IUT. This method groups the inputs of the specification into disjoint sets like basic inputs, level-1 inputs, level-2 inputs and so on and tests them in that order. The *Basic subgraph* contains edges from *basic inputs. Level-i* subgraph contains inputs from *basic subgraph, level-1 inputs* etc., up to *level-i inputs.* The *basic subgraph* should have the following properties: (i) it is a Strongly Connected Spanning Subgraph (SCSS) of the specification (ii) it has the minimum number of inputs from the inputs of the specification and (iii) all the states in the basic subgraph have a SIS. The *basic subgraph* should be tested by a method which has better fault detection capability even in the presence of multiple faults in the IUT. Of all the fault detection methods, the R-method has better fault detection capability and it is used for testing the *basic subgraph.* For testing *level-i* inputs, the preamble and postamble should be used from the *level-i subgraph* which are already tested. If the preamble and postamble are error-free and testing starts from the state specified by the test sequence, fault diagnosis is possible for *level-i* inputs. Whenever an error is detected, homing and error recovery subsequences are applied before further testing can be continued. Multilevel method is described in detail in [Kripanandan (1994), Neelakantan (1994)].

7 COMPARISON OF TESTING METHODOLOGIES

If the IUT has a reliable reset in all the states and the SIS of the specification is also the SIS of the IUT, D, W and U methods can detect the faulty IUT. The test result analysis does not give the exact location of faults, as after the SIS verification phase, all the edges used in the SIS may not be error-free. As the test sequences are optimized, if any error occurs, it is not possible to distinguish between whether the error is due to a transfer fault in the previous transition or due to an output fault in the current edge under test. Also one fault may mask the presence of another fault in the IUT [Miller (1992)]. In general, the W-method will produce the longest test sequence and the U and D-methods will produce shorter test sequence of comparable lengths [Sidhu (1989)].

Fault coverage : The fault coverage or the effectiveness of the test sequence depends upon three factors: (i) for each edge in the specification the testing should start at the state specified by the test sequence. (ii) the preamble should be error-free and (iii) all the edges in SIS (postamble) should also be error-free. In the M-method, if the basic subgraph is error-free, all the three conditions are satisfied [Neelakantan (1994)]. The DA method assumes an error-free preamble for all the edges and errors can be present in the postamble. Hence to locate faults, additional testing is necessary. The CSP method can locate a single transfer fault easily. However, when multiple faults occur at successive edges, locating these faults can become difficult.

Number of subgraphs : In all the fault detection and fault diagnostic methods, the entire specification is taken for testing purposes. On the other hand, the M-method, splits the specification into different subgraphs and tests them one after another. In general D, W and U methods can be considered as special cases of the M-method where it has only two subgraphs namely the *basic subgraph* and *level 1 subgraph.* The edges in the SIS and state verification subsequence of the D, W and U methods can be considered as the basic subgraph. The transition verification phase can be considered as the level 1 subgraph. In D, W and U method, if the IUT succeeds the SIS verification phase, it does not guarantee that all the edges in the SIS are error-free. Whereas in the M-method, if *basic subgraph* succeeds the test, it guarantees that preamble and SIS are error-free.

Length of the test sequence : The length of the test sequence generated based on the FR approach will be comparable to D, W and U methods, as it assumes that only one fault can be present in the IUT. The length of test sequence based on the DA method increases with the number of faults present in the IUT and it will be comparable to the D, W and U method if there are no errors in the IUT. As the SIS of the *basic subgraph* of M-method is usually larger than the SIS used in other methods (D, W and U method), the length of test sequences of the M-method will be larger than the corresponding test sequence based on D, W, and U Method.

Error Reporting : Test result analysis and some cases, extra test sequences are required to identify the erroneous transitions in FR, DA, CSP methods, whereas in the M-method, the erroneous transitions are reported once they are detected.

Error recovery : The DA, FR, CSP methods uses *reliable reset* for error recovery in the IUT. On the other hand, the M-method uses homing and recovery sequences for error recovery.

Applicability to real life protocols : If *reliable reset* is present in the IUT, the DA, CSP and M methods are more attractive. If reliable reset is not present, the M-method can be used. If there is no basic subgraph which is error-free, the M-method may not be applicable. In that case, the DA method may be applicable if all the faults in the IUT are *directly reachable.*

8 SUMMARY AND CONCLUSION

In this paper we saw different aspects of protocol conformance testing. The testing methodology and framework provided by ISO has resulted in the development of standard test architectures and testing techniques and they were briefly explained. Different test sequence generation methods based on FSM models were also discussed. The testing methods can be grouped broadly into two categories: Fault detection methods, whose aim is to detect whether the IUT is faulty; Fault diagnostic methods, whose aim is to identify the erroneous transitions. We compared the test sequence generation methods in terms of its fault coverage, the availability of reliable reset and length of the test sequence. In general, the aim of all the test sequence method is to get minimal length sequence. But optimization introduces an undesirable side effect of reduced fault coverage. Thus, during test sequence generation effectiveness (fault coverage) should be given preference over the efficiency (length of test sequence). Most of the assumptions made by testing methods like completely specified, reliable reset, self loop at each node are not satisfied by most of the real life protocols. The possibility of overcoming these barriers should be explored.

REFERENCES

Aho A.V., Anton T. Dahbura, David Lee, and M. Umit Uyar (1991) An optimization technique for protocol conformance test generation based on UIO sequences and rural chinese postman tours. *IEEE Trans. on Communication*, 39:11, 1604-1615.

Bochmann G.V., R. Dssouli, A. Das, M. Dubuc, A. Ghedamsi, and G. Luo (1991) Fault models in testing. In *Protocol Test Systems*, pages 17–32.

Bochmann G.V., and A. Ghedamsi (1993) Multiple Fault Diagnostic Tests for Finite State Machines. In *IEEE Infocom 93*, pages 782–791.

Bosik B.S. and Uyar M.U. (1991) FSM based formal methods in protocol conformance testing : from theory to implementation. *CN and ISDN Systems*, 22:7–33.

Boyd S.C. and Hasan Ural (1991) On the complexity of generating optimal test sequences. *IEEE Transactions on Software Engineering*, 17:976–979.

Chen M, Choi Y., and Kershenbaum A. (1990) Approaches utilizing segment overlap to minimise test sequences. In *Protocol Spec. Testing, and Verification X*, pages 85–98.

Chow T. S. (1978) Testing software design modeled by finite state machines. *IEEE Transactions of Software Engineering*, SE-4:178–187.

Chan W. Y. L., S. T. Vuong, and M. R. Ito (1989) An improved protocol test generation procedure based on UIOS. In *SIGCOMM'89*, pages 283–294.

Fujiwara S., G. V. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi (1991) Test selection based on finite state models. *IEEE Trans. on Software Engineering*, 17.

Gonenc G. (1970) A method for the design of fault detection experiments. *IEEE Transactions of computers*, 19:551–558.

ISO (1989) *Information technology -OSI conformance testing methodology and framework – Part 1: General concepts.* International Organization for Standardization.

ISO (1989) *Information technology -OSI conformance testing methodology and framework – Part 2: Abstract test suite specification.* International Organization for Standardization.

ISO (1989) *Information technology -OSI conformance testing methodology and framework – Part 3: The Tree and Tabular Combined Notation.* International Organization for Standardization.

ISO (1989) *Information technology -OSI conformance testing methodology and framework – Part 4: Test realization.* International Organization for Standardization.

ISO (1989) *Information technology -OSI conformance testing methodology and framework – Part 5: Requirements on test laboratories and clients for the conformance assessment process.* International Organization for Standardization.

Linn R J. (1989) Conformance evaluation methodology and protocol testing. *IEEE Journal on selected areas in communications*, 7(7):1143–1158.

Kohavi Z. (1978) *Switching and Finite Automata Theory.* McGraw Hill.

Kripanandan R. S. and S. V. Raghavan (1994) Multilevel approach to protocol conformance testing. to appear in *CN and ISDN Systems.*

Kripanandan R. S. (1990) Multi-level approach to protocol conformance testing. Master's thesis, Department of Computer Science, Indian Institute of Technology Madras.

Lombardi F. and Y.N. Shen (1992) Evaluation and improvement of fault coverage of conformance testing by UIO sequences. *IEEE transactions on communications*, 40,8:1288–1293.

Miller R.E. and Sanjoy Paul (1992) Structural analysis of a protocol specification and generation of a maximal fault coverage conformance test sequence. Technical report, University Of Maryland.

Miller R.E. and Sanjoy Paul (1993) On the generation of minimal length conformance tests for communication protocols. *IEEE transactions on networking*, 1:116–129.

Neelakantan B. and S.V. Raghavan (1994) Scientific Approach to Multilevel Method. Communiated to *IEEE transactions on networking.*

Naito S. and M. Tsunoyama (1981) Fault detection for sequential machines by Transition tours. In *Proceedings of Fault tolerant computing systems*, pages 238–243.

Rayner D. (1987) OSI conformance testing. *CN and ISDN Systems*, 14:79–98.

Ramalingam T., Anindya Das, and K. Thulasiraman (1992) On conformance test and fault resolution of protocols based on FSM model. In *Networks 92*, pages 435–475.

Sarikaya B. and G.V. Bochmann (1982) Some experience with test sequence generation for protocols. In *Protocol Specification, Testing, and Verification II*, pages 555–567.

Sabnani K. and Anton Dahbura (1988) A protocol test generation procedure. *CN and ISDN Systems*, 15:285–297.

Sidhu D.P. (1990) Protocol testing: The first ten years, the next ten years. In *Protocol Specification, Testing, and Verification X*, pages 47–68.

Sidhu D.P. and Tingkau Leung (1989) Formal methods for protocol testing: A detailed study. *IEEE Transactions of Software Engineering*, 15(4):413–426.

Shen Y. N., F. Lombardi, and A. T. Dabhura (1989) Protocol Conformance Testing using MUIO sequences. In *Protocol Spec., Testing, and Verification IX*, pages 131–143.

Sidhu D.P. and Raghu Vallurupalli (1990) On arbitrariness in protocol conformance test generation. Technical report, University Of Maryland.

Ural H. and K Zhu (1993) Optimal length test sequence generation using distinguishing sequences. *IEEE transactions on networking*, 1:358–371.

Vuong S.T. and Kai C. Ko (1990) A novel approach to protocol test sequence generation. In *Globalcom'90*, pages 1880–1884.

Yao M., A. Petrenko, and G V Bochmann (1993) Conformance testing of protocol machines without reset. Technical report, University Of Montreal.

Yang Bo and Hasan Ural (1990) Protocol conformance test generation using multiple UIO sequences with overlapping. In *SIGCOMM'90*, pages 118–125.

Zeng H. X., S. T. Chanson, and B. R. Smith (1989) On Ferry clip approaches protocol testing. *CN and ISDN Systems*, 17:77–88.

9 BIOGRAPHY

B. Neelakantan received his B.E degree in Computer Science and Engineering from Thiagarajar College of Engineering Madurai, India in 1992. He is currently working towards his MS (by Research) degree at the Department of Computer Science and Engineering, Indian Institute of Technology, Madras. His research work is supported by Government of India research scholarship. His research interests are networks, protocol and graph theory.

S.V. Raghavan is on the faculty of the Department of Computer Science and Engineering, Indian Institute of Technology, Madras. He is also the Chief Investigator of the project on *Education and Research in Computer Networking* jointly sponsored by the Department of Electronics, Government of India and the United Nations Development Programme. He is a life member of the Computer Society of India, a member of the Institution of Engineers and a fellow of institute of Electronics and Telecommunication Engineers. He is presently serving on the Board of Editors of the journal of IETE for computers and control. He is also a member of the Editorial Advisory Board for Computer Communications, Butterworth-Heinemann Ltd. His research interests are networks, protocols, multimedia systems and performance.