THE STATE OF THE ART IN TESTING PROTOCOL IMPLEMENTATIONS

D. Rayner

Rapporteur: Mr. P. Ezhilchelvan

The State of the Art in Testing Protocol Implementations

by D. Rayner
Leader of the Protocol Standards Group
Division of Information Technology and Computing
National Physical Laboratory
Tedington, Middx. TW11 0LW

ABSTRACT

International standards for Open Systems Interconnection (OSI) services and protocols are well advanced. Complementary standardization work has begun for testing conformance of products to OSI protocol standards. This is drawing upon about 6 years research and development work on techniques and tools for testing protocol implementations. This presentation reviews the areas of general agreement: the meaning of conformance; abstract test methods; and test suite design principles. It also reviews the diversity of approaches used for designing systems to implement those agreed abstract test methods. These more mature areas are given emphasis, but further study areas are also identified.

0. Introduction

Open Systems Interconnection (OSI) will not be completely achieved until systems can be tested in order to determine whether they conform to the relevant OSI protocol standards.

Research groups have been developing techniques and systems for testing protocol implementations for the past 6 years. Much of this work has been conducted with either formal or informal international collaboration. These techniques and systems are now being used by suppliers, carriers, third party test centres and a few users. This has improved the compatibility between different systems using and providing common services, such as Teletex, and enabled multi-vendor demonstrations of Open Systems Interconnection (OSI) protocols to take place. It has also led to recent rapid progress towards a standard for "OSI conformance testing methodology and framework" and the development of proposed standard test suites for X.25 and Teletex protocols. Standard test suites for OSI protocols will follow. This will facilitate the comparison of results of testing conducted by different organisations.

The subject is now mature enough for the aspects of general agreement to be included in a university course on network protocols. This paper covers the areas which are most suitable for inclusion in such a course and presents them using the terminology used in the current working draft of the conformance testing standard. This common terminology is recommended in preference to the wide variety of terminology used by various research groups.

The main problem in presenting this subject in a university course is finding a suitable text book. All that can be suggested is that the North Holland series of books called "Protocol Specification, Testing and Verification" are used. They give the proceedings of the IFIP international workshop series on this subject and are a very good source of papers on both protocol testing and on the use of formal description techniques. The 5th workshop proceedings will include a paper similar to this one, by the same author, also based on the February 1985 working draft [1] of the conformance testing standard.

1. Conformance Requirements

A prerequisite of ISO's work on OSI conformance testing was an understanding of the meaning of conformance in the context of OSI. Firstly it was decided that the term should be restricted to the conformance of an implementation or system to one or more protocol standards. It is incorrect to claim either conformance to the reference model or to service standards. Thus any conformance requirements related to the service must be stated in the relevant protocol standard(s).

The study of conformance revealed that there was much misunderstanding of the matter by protocol definers. In particular, it is important to distinguish two different types of conformance requirement: static and dynamic conformance requirements. Dynamic ones define the allowed behaviour in instances of communication, i.e. what a protocol implementation does; they constitute the bulk of a protocol specification. Static ones, on the other hand, define the allowed minimum capabilities of an implementation, i.e. what a protocol implementation contains; they are often only found in special conformance clauses.

Another important distinction is between mandatory requirements, conditional requirements, options and prohibitions. Each of these can apply to either static or dynamic conformance so it is important that protocol standards should state which is which. For example, it is possible to have a protocol procedure whose use in a particular instance of communication is optional but whose support by an implementation is mandatory, or indeed vice versa.

There may also be requirements on what a supplier or implementor should state about an implementation. In any case, for the purposes of conformance testing, a statement is needed of the capabilities and options which have been implemented. Such a statement is called a protocol implementation conformance statement (PICS). It should be consistent with the static conformance requirements in that it should include all mandatory capabilities plus all additional capabilities required as a result of the options which are supported.

All these concepts are covered in a draft answer to the question of conformance [2] which is being voted on as an approved interpretation of the OSI reference model. In addition, a checklist has been produced [3] to assist protocol definers to get a clear unambiguous statement of the desired conformance requirements. This was based largely on the UK input of a published conference paper [4]. It is expected to be progressed as an annex to the conformance testing methodology and framework standard [1].

2. Types of Testing

In principle, the objective of conformance testing is to establish whether the implementation being tested conforms to the specification in the relevant standard. Practical limitations make it impossible to be exhaustive, and economic considerations may restrict testing still further.

Four types of conformance testing have been identified, according to the extent to which they provide an indication of conformance:

- basic interconnection tests, which provide prima facie evidence that an implementation under test (IUT) conforms;

- functional range tests, which determine the capabilities of an IUT, i.e. which features and options are supported;

- conformance tests, which endeavour to provide as comprehensive testing

as possible over the full range of requirements specified by the standard;

- conformance resolution tests, which provide a definite yes/no answer in the context of specific conformance issues.

2.1 Basic Interconnection Tests

These provide limited testing of the main features in a standard, to establish that there is sufficient conformance for interconnection to be possible, without trying to perform thorough testing.

Basic interconnection tests are appropriate:

- for potentially detecting severe cases of non-conformance;
- as a first filter before undertaking more costly tests;
- to give a prima facie indication that an implementation which has passed full conformance tests in one environment still conforms in a new environment (e.g. in a multi-layer implementation, to check that a tested (N-1)-implementation has not undergone any severe change when linked to the (N)-implementation);
- for use by users of implementations, to determine whether the implementations are usable for communication at all with other conforming implementations, e.g. as a preliminary to data interchange.

Basic interconnection tests are inappropriate:

- as a basis for claims of conformance by the supplier of an implementation;
- as a means of arbitration to determine causes for communications failure.

Basic interconnection tests should be standardized as a very small subset of a full conformance test suite.

2.2 Functional Range Tests

Functional range tests determine which options and features are supported by an IUT at various levels of detail, from which major subsets or classes are supported to the range of values supported for a particular parameter. They can include not only tests to determine which protocol data units (PDUs) are supported, but also ones to determine which service primitives are supported.

Functional range tests can be used to:

- check the validity of the protocol implementation conformance statement made for an IUT;
- check that the static conformance requirements are met;
- enable an efficient selection of other conformance tests to be made for a particular IUT.

Functional range tests should be standardized as a subset of a full conformance test suite.

2.3 Conformance Tests

Conformance tests are intended to provide as thorough testing of an implementation as is practical, over the full range of requirements specified in a standard. Since the number of possible combinations of events and timing of events is infinite, such testing cannot be exhaustive. There is a further limitation, namely that these tests are designed to be run collectively in a single test environment, so that any faults which are difficult or impossible to detect in that environment can be missed. Therefore, it is possible that a non-conforming implementation passes the conformance test suite; one aim of the test suite design is to minimise the number of times that this occurs.

It is reasonable to regard an implementation as conforming if it satisfies the conformance test suite, so long as there is no evidence to the contrary.

Conformance tests are appropriate:

- as a basis for claims of conformance, so long as other tests or live usage have not revealed contrary evidence;

- as a basis for procurement.

Conformance tests are inappropriate:

- for resolution of problems experienced during live usage or where other tests indicate possible non-conformance even though the conformance test suite has been satisfied.

Conformance tests should be standardized.

2.4 Conformance Resolution Tests

These provide diagnostic answers, as near to definitive as possible, to the resolution of whether an implementation satisfies particular requirements. Because of the problems of exhaustiveness noted above, the definitive answers are gained at the expense of confining tests to a narrow field.

The test architecture and test method will normally be chosen specifically for the requirements to be tested, and need not be ones that are generally useful for other requirements; they may even be ones that are regarded as being unacceptable for generally specified conformance tests, e.g. involving implementation-specific methods using, say, the diagnostic and debugging facilities of the specific operating system.

The distinction between conformance tests and conformance resolution tests may be illustrated by the case of an event such as a Reset. The conformance tests may include only a representative selection of conditions under which a Reset might occur, and may fail to detect incorrect behaviour in other circumstances. The conformance resolution tests would be confined to conditions under which incorrect behaviour was already suspected to occur, and would confirm whether or not the suspicions were correct.

Conformance resolution tests are appropriate:

- for providing a yes/no answer in a strictly confined and previously identified situation (e.g. during implementation development, to check whether a particular feature has been correctly implemented, or during operational use, to investigate the cause of problems);

- as a means for identifying and offering resolutions for deficiencies in a

current conformance test suite.

Conformance resolution tests are inappropriate:

- as a basis for judging whether or not an implementation conforms overall;
- as a condition for procurement.

Conformance resolution tests need not be standardized.

2.5 Other Types of Testing

In addition to conformance testing, other types of testing have been proposed [5] including:

- performance tests to measure the performance characteristics of an IUT, such as its throughput and responsiveness under various conditions;
- robustness tests to determine how well an IUT recovers from various error conditions.

There may be some overlap between these types of test and conformance tests and where there is then such tests could be extracted as a subset of a full conformance test suite. However, where the test purpose is not concerned with conformance, then such tests fall outside the current scope of standardization.

3. Test Suite Design

The ISO framework for conformance testing includes guidance on the design of conformance test suites. A test suite will be structured into test groups, possibly substructured into subgroups. These will consist of a set of related abstract test specifications, each with its own test purpose, which should be precise and detailed so that it is clear what failure would mean. There will also be appropriate instructions for determining how to perform test selection and decide on the order in which the tests should be run.

Each abstract test specification will consist of a tree of test events (e.g. the sending or receipt of a service primitive or PDU) structured into test steps. A test step may be shared by many abstract test specifications. In order to run a test, the abstract test specification has first to be converted into an executable test definition appropriate for some specific testing system; for the tests are standardized at an abstract level, independent of any particular way of executing them.

A full conformance test suite, for a particular protocol, should be capable of testing all mandatory and optional features over the maximum range of parameters and variations.

The design of a test suite must take a number of factors into consideration, and should result in the optimum combination of them. The factors relevant for single-layer conformance test suites are itemized below.

3.1 Features

A full test suite should be capable of testing all the mandatory and optional features of the protocol, selected in all the feasible combinations allowed by the static conformance requirements of the protocol specification. If the specification allows for selecting a partial set of features, such as 'receive only', the test suite must be adaptable to this situation.

3.2 Protocol Phases

A full test suite will allow the selection, for each function of the protocol, of specific tests of the major protocol phases, in various combinations:

- connection establishment, when a connection is being set up;
- data transfer phase, while a connection is current (or at all times for a connectionless protocol);
- connection release, when a connection is being broken.

These phases are those used in OSI service and protocol descriptions. There may be subphases (half-open connection, data transfer awaiting acknowledgement, non-connection requests in the idle state, etc.).

3.3 Variations

The full test suite must include a range of variations in the following domains:

- sequence variations (the order in which PDUs occur);
- timing/timer variations;
- PDU encoding variations;
- parameter variations in PDUs.

Possible choices are suggested below:

- support of all PDU types and all structural variations of each type;
- 'normal' or default values for each parameter on each PDU;
- boundary values plus at least one mid-range value for each integer parameter;
- for bitwise parameters, as many values as is practical, but not less than all of the 'normal' or common values;
- for interdependent pairs of PDU parameters, 'critical' value pairs (representing multi-dimensional boundaries) and one 'normal' value pair;
- all sequences of PDUs where one PDU is 'out of sequence' with respect to the defined protocol;
- at least one invalid PDU type;
- at least one invalid value for each PDU parameter, where such invalid values exist;

- all defined timers should be exercised, i.e. allowed to expire at least once.

3.4 Valid/Invalid Behaviour

A test suite will check that the behaviour of the IUT is valid under all the above variations. Invalid and unexpected outputs from the IUT must be detected and given a specific branch in the abstract test specification to end the test and be identified.

Consistent reaction of the IUT upon receipt of invalid PDUs from the tester must also be exercised. Single erroneous PDUs should be mapped into correct (N-1)-service primitives.

Invalid PDUs are obtained by extending the variations (3.3) beyond the valid ranges.

3.5 Interdependence

The design of a test suite must consider test interdependence. Some tests must be successfully completed before others are attempted. For example, before testing detailed functional behaviour related to data transfer, the tests of connection establishment and transition to the data transfer phase should be successfully carried out.

Another type of dependency is the inter-functional dependency. Before selection of tests of data transfer, the capability to negotiate data length must be tested and established.

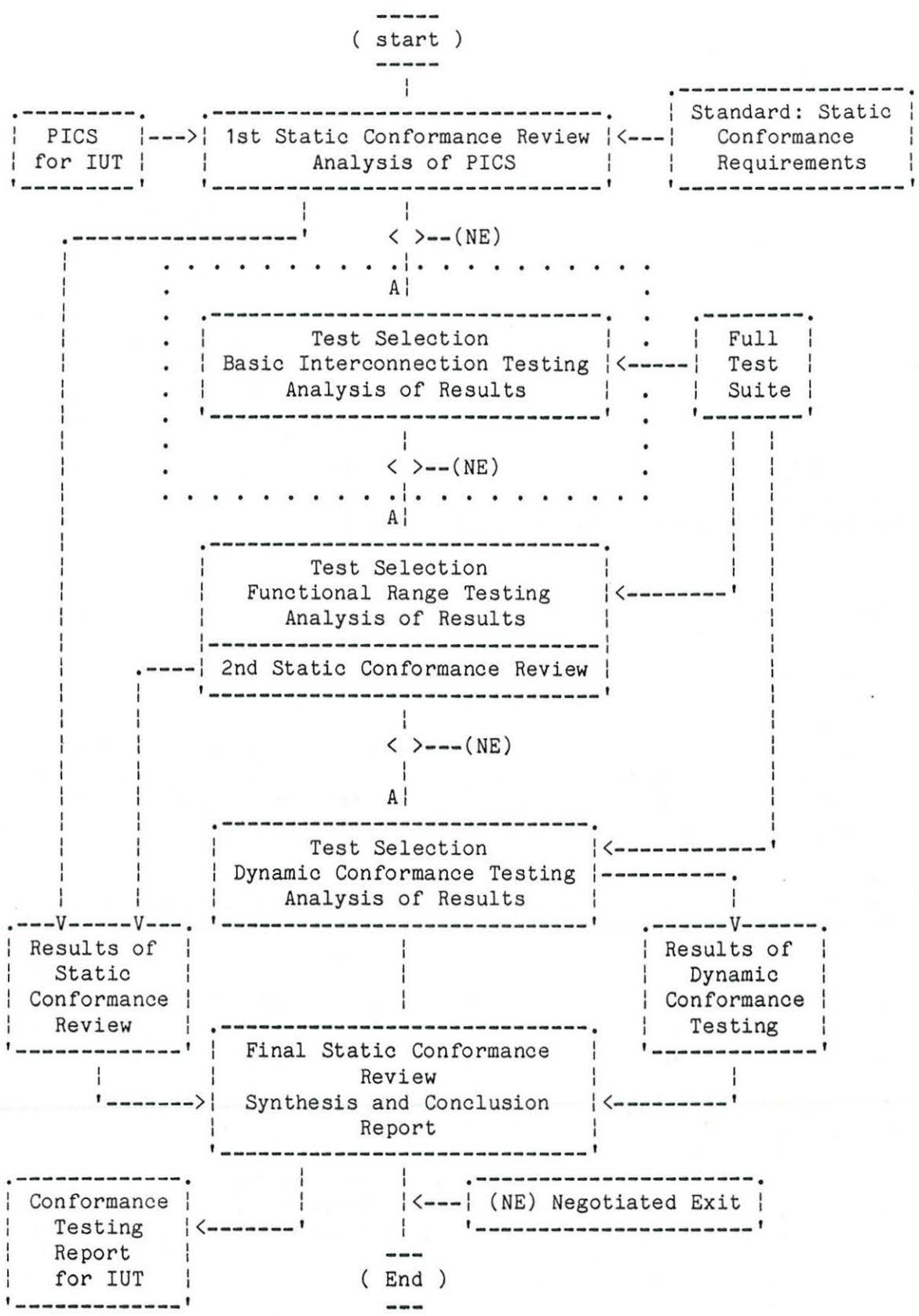
4. Typical Use of a Conformance Test Suite

It is important to understand how conformance testing should relate to static and dynamic conformance requirements and to the protocol implementation conformance statement. There are many possible ways of interleaving dynamic conformance testing with reviews of the information obtained about static conformance which can then affect subsequent test selection. However, ISO has illustrated the typical use of a conformance test suite in the form of a flow-chart, to clarify the main interrelationships. This is shown in Figure 1.

The first step, static conformance review, is a paper analysis, in which the protocol implementation conformance statement accompanying the IUT will be analysed for its own consistency, and its consistency with the static conformance requirements specified in the protocol standard(s) to which the IUT is claimed to conform. This paper analysis can be followed by one or two steps of live testing. The first is basic interconnection testing which is optional, and would be used detect severe cases of non-conformance and to determine whether more extensive testing is going to be worthwhile. The second is functional range testing, which will ascertain the validity of the PICS with respect to the actual, observable capabilities of the IUT.

A second static conformance review will combine the results of the functional range tests with the results of the first review.

The fourth step, dynamic conformance testing, will concentrate on live testing, checking the correct behaviour of the protocol implementation. State transition control, syntactic checking of the protocol elements, and behaviour of the implementation are in the scope of this phase. The behaviour will be tested in various instances of communication, both simple and complex, independent and



NOTE: A: Acceptable Results

Figure 1. Typical use of a conformance test suite

dependent on the environment, normal and erroneous.

It cannot be proved by testing that an implementation conforms dynamically in all instances of communication. However, it can be shown by testing that an implementation consistently conforms dynamically in representative instances of communication.

Two phases of 'paper analysis' will take place, one before and the other after the live testing phase. The first will be a phase of preparation/selection of the test suite(s) to be used, based on the results of the second static conformance review. The second will be a phase of analysis of the results of the live testing from the dynamic conformance point of view.

The final static review involves a synthesis of the results of the dynamic conformance tests with those of the second static conformance review. A conclusion on the conformance of the IUT to the requirements of the standard(s) can then be reached. This conclusion is recorded in a Conformance Testing Report detailing the tests run, their results, and the environmental conditions.

Provisions for "negotiated exits" can be seen in Figure 1. They are points where the concerned parties can decide that the results of the previous step are not good enough to justify continuing the tests.

5. Abstract Testing Methodology

5.1 Control and Observation and Abstraction

The ISO abstract testing methodology is based upon the OSI reference model. Abstract test methods are described in terms of what outputs from the entity under test are observed and what inputs to it can be controlled. More specifically, an abstract test method is described by identifying the points closest to the entity under test at which control and observation are to be exercised.

The OSI protocol standards define allowed behaviour of a protocol entity (i.e. the dynamic conformance requirements) in terms of the PDUs and both the abstract service primitives (ASPs) above and below that entity. Thus the behaviour of an (N)-entity is defined in terms of the (N)-ASPs and (N-1)-ASPs (the latter including the (N)-PDUs). Each of these two sets of interactions could be observed and controlled from several different points, directly or remotely, as identified in Figure 2.

Note that the possible points of observation and control are identified by three factors:-

(a) whether it is the ASPs or PDUs which are observed and controlled;
(b) the layer identity of the ASPs or PDUs concerned;
(c) whether they are controlled and observed within the system under test or in a system remote from the system under test - if the latter then the ASPs or PDUs are distinguished by the addition of a double-quote character (").

Complete control and observation of the (N-1)-ASPs (or (N-1)-ASP"s) will include control and observation of the (N)-PDUs (or (N)-PDU"s), but not vice versa.

It is possible that the (N)-ASP activity of the entity under test might not be controllable nor observable, directly or indirectly, in which case this activity is said to be hidden. It is, however, assumed that the (N-1)-ASP activity will at least be indirectly observable and controllable, via some real means of communication (e.g. via (N-1)-ASP"s). Thus when the (N-1)-ASPs are not

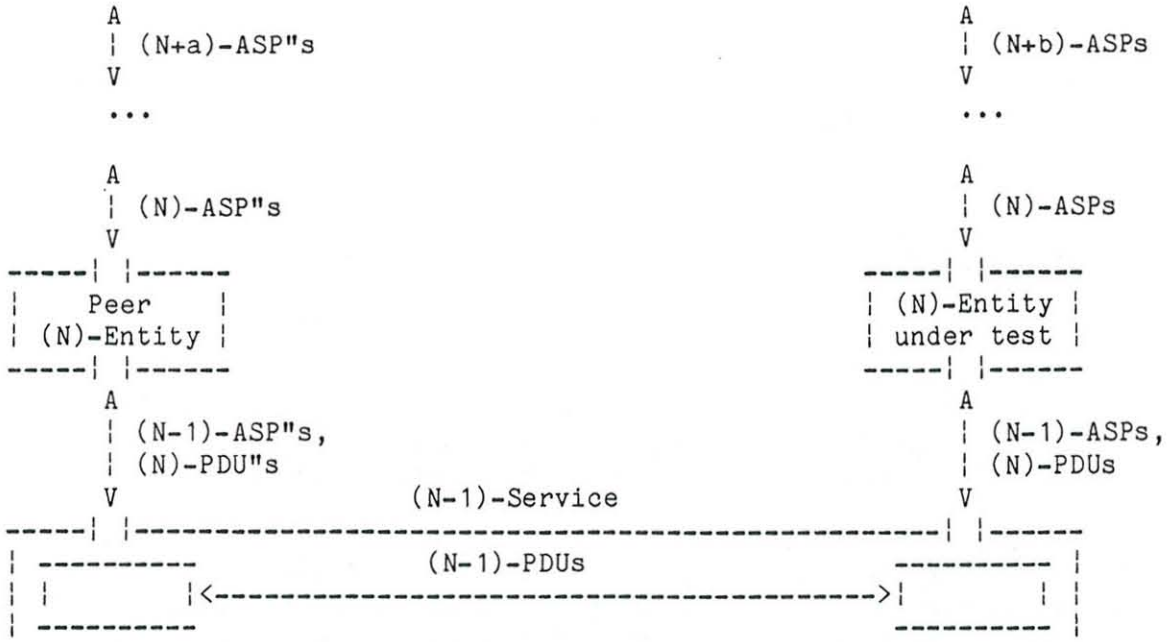


Figure 2. Possible points of control and observation

controllable nor observable directly, conformance testing can only be carried out if the (N-1)-service is provided sufficiently reliably for control and observation to take place remotely.

It is important to make the distinction between the interactions which are controlled and observed, and the subset of those observations on which judgement is passed. Judgement can only be passed on whether or not the conformance requirements are being met. This will usually apply to the PDUs exchanged rather than the particular realisations of ASPs. However, the reason for wishing to control ASPs is to try to determine more precisely which PDUs should be exchanged in a particular test.

The virtue of expressing control in terms of ASPs is that this is an abstract form of specification which does not unduly limit the freedom of testers to implement the tests in different ways using whatever interfaces are accessible externally, provided that the required degree of control and observation is available.

Associated with this abstract view of control and observation, there needs to be a correspondingly abstract view of the testers which perform the control and observation. They are referred to as the lower and upper testers. The lower tester is the means for providing control and observation of events which approximate to what the IUT sees at its lower SAP (i.e. the (N-1)-ASPs, (N-1)-ASP"s, (N)-PDU"s, etc). The upper tester is the means for providing control and observation of events which approximate to what the IUT sees at its upper SAP (i.e. the (N)-ASPs, (N+1)-ASPs, etc). In addition, there is the concept of test coordination procedures which are the rules for cooperation between upper and lower testers during testing.

5.2 Test Method Overview

Conformance test suites for a given OSI protocol or set of protocols should be defined for each of a limited number of abstract test methods. Testers would then be expected to select the most appropriate suite for the IUT or system under test (SUT). The chosen abstract test methods fall into three main categories:-

(1) Local test methods: which use control and observation of the ASPs directly above and below the IUT;

(2) Distributed test methods: which use control and observation of the (N-1)-ASP"s together with control and observation of the ASPs directly above the IUT;

(3) Remote test methods: which use control and observation of the (N-1)-ASP"s, with the ASP activity directly above the IUT being hidden.

The abbreviated name for an individual abstract test method begins with an 'L', 'D' or 'R', to denote Local, Distributed or Remote respectively, according to the main category to which it belongs. These categories can be applied to testing a single-layer at a time, in which case the second letter of the abbreviated test method name is an 'S', or they can be applied to testing all layers of a multi-layer IUT, in which case the second letter of the abbreviated name is an 'M'. Finally, single-layer test methods are qualified by the adjective 'embedded' when they apply to a single-layer within a multi-layer IUT, such that the control and observation is less direct than it would be for the corresponding method of testing a single-layer IUT; for such methods an 'E' is added as a third letter in the abbreviated name.

With this naming scheme for abstract test methods, the following methods are proposed:-

LS - Local single-layer test method
LM - Local multi-layer test method
DS - Distributed single-layer test method
DSE - Distributed single-layer embedded test method
DM - Distributed multi-layer test method
RS - Remote single-layer test method
RM - Remote multi-layer test method

5.3 The Local Single-layer Test Method

The LS test method is illustrated in Figure 3.

The tests defined for this method would provide the base from which corresponding tests for any other abstract test method could be derived. They also provide a basis for deriving multi-layer tests.

They are useful to implementors for "unit testing", that is the testing of an implementation of a protocol entity in isolation from the rest of the system.

However, since this method involves no communication over an (N-1)-service-provider, the protocol entity may require further testing using another method which uses an (N-1)-service-provider. It may, however, be satisfactory to carry out a conformance test suite using the LS test method and then follow it up with a confidence check by running a basic interconnection test suite using the DS, DSE or RS test method.

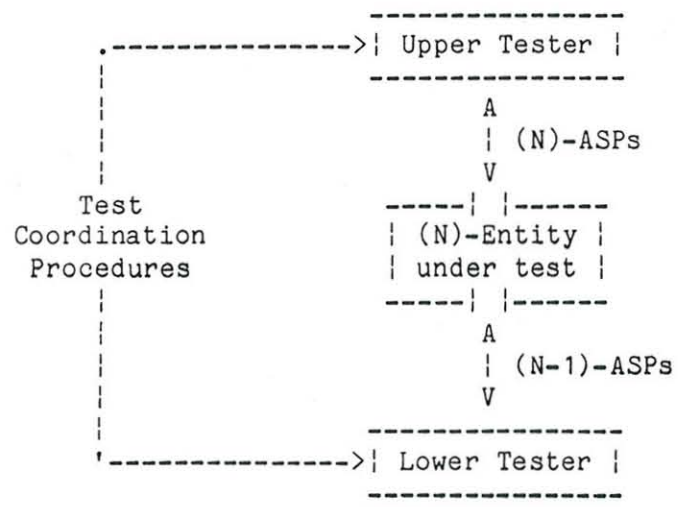


Figure 3. The local single-layer test method

5.4 The Distributed Single-layer Test Method

The DS test method is illustrated in Figure 4.

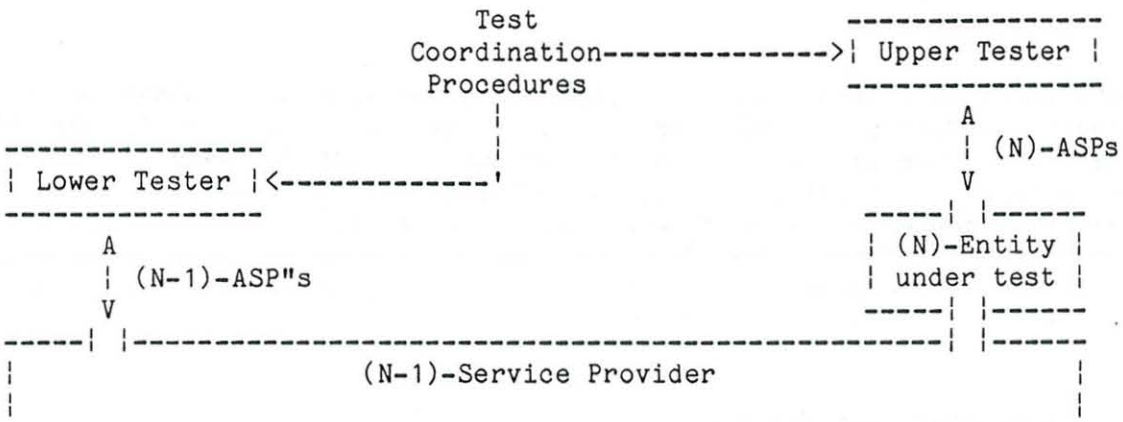


Figure 4. The distributed single-layer test method

When applicable, this method provides the most complete form of single-layer testing over the (N-1)-service-provider. It permits a large number of (N)-PDU errors and unusual but valid (N-1)-ASP"s to be included in the tests.

The tests defined for this method could be realised in practice in many different ways, provided that the system under test can be made to provide some realisation of the (N)-ASPs. However, they would not constrain the form of implementation of the upper and lower testers.

The upper tester will not be purely passive since it needs to exercise some control over which (N)-ASPs are generated, but it will not have the same scope for introducing errors as the lower tester.

Higher-layer forms of describing the tests (e.g. in terms of (N)-ASP"s) should

be used where this will reduce the size of the test specification without loss of required precision. This will enable identification to be made of those tests which are equivalent to ones for a method involving control and observation of (N)-ASP"s instead of (N-1)-ASP"s. Those who wish to use this subset of the test suite should be warned that it is not adequate as a complete conformance test suite; although it could be used as a set of basic interconnection tests. The problem is that not all (N-1)-ASP"s can be invoked merely by control of (N)-ASP"s; e.g. Network Reset cannot be invoked by control of Transport Service Primitives.

5.5 The Remote Single-layer Test Method

The RS test method is illustrated in Figure 5.

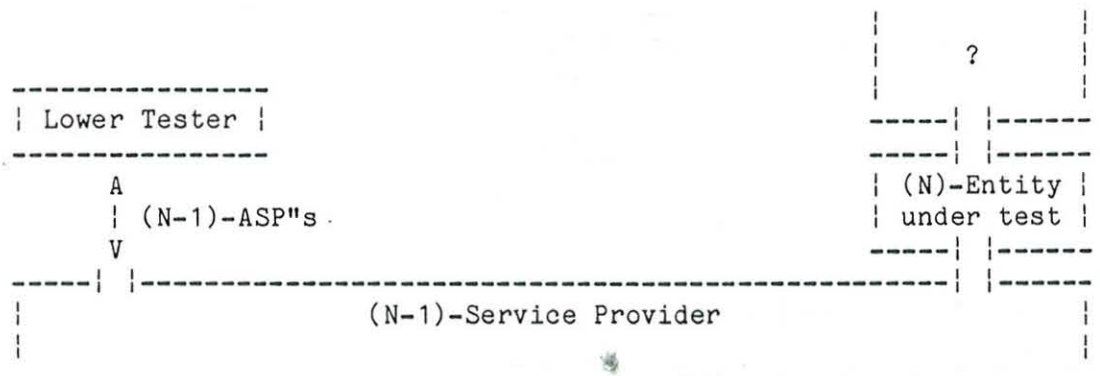


Figure 5. The remote single-layer test method

Tests for this method should be possible to use with all systems under test. It places no additional requirements for the sake of testing on the implementation other than those placed on it by the protocol standard. (N-1)-ASP"s are used, rather than just (N)-PDU"s, in order to allow tests to specify when the underlying connection should be set up or released.

This method is appropriate for those systems in which control and observation of (N)-ASPs is not possible.

5.6 Multi-layer Test Methods

Multi-layer testing consists of testing all the layers of a multi-layer IUT as a whole, without accessing the inter-layer interfaces within the IUT. This type of method could be used when these inter-layer interfaces are not accessible, and also when the combined allowed behaviour of the multi-layer implementation must be known.

The LM test method is similar to the LS test method but the IUT goes from (N-1)-ASPs up to (N+n)-ASPs and therefore the upper tester controls and observes (N+n)-ASPs. The same transformation turns the DS test method into the DM test method. In terms of control and observation, the RM test method is the same as the RS test method. However, in all three cases, the lower tester will need to deal with (N)-PDU"s up to (N+n)-PDU"s.

5.7 Incremental Embedded Single-layer Testing

Incremental embedded single-layer testing permits testing of multi-layer implementations layer by layer, from (N) up to (N+n), without requiring the access to interfaces for each layer within the implementation. This is applicable to both distributed and remote test methods. The distributed approach is illustrated in Figure 6. For a general layer (N+i) the DSE method uses control and observation of (N+i-1)-ASP"s and (N+n)-ASPs. Note that for the top layer in the multi-layer IUT, (N+n), the ordinary DS test method is used.

In the same way, the RS test method can be used incrementally for each layer of the IUT.

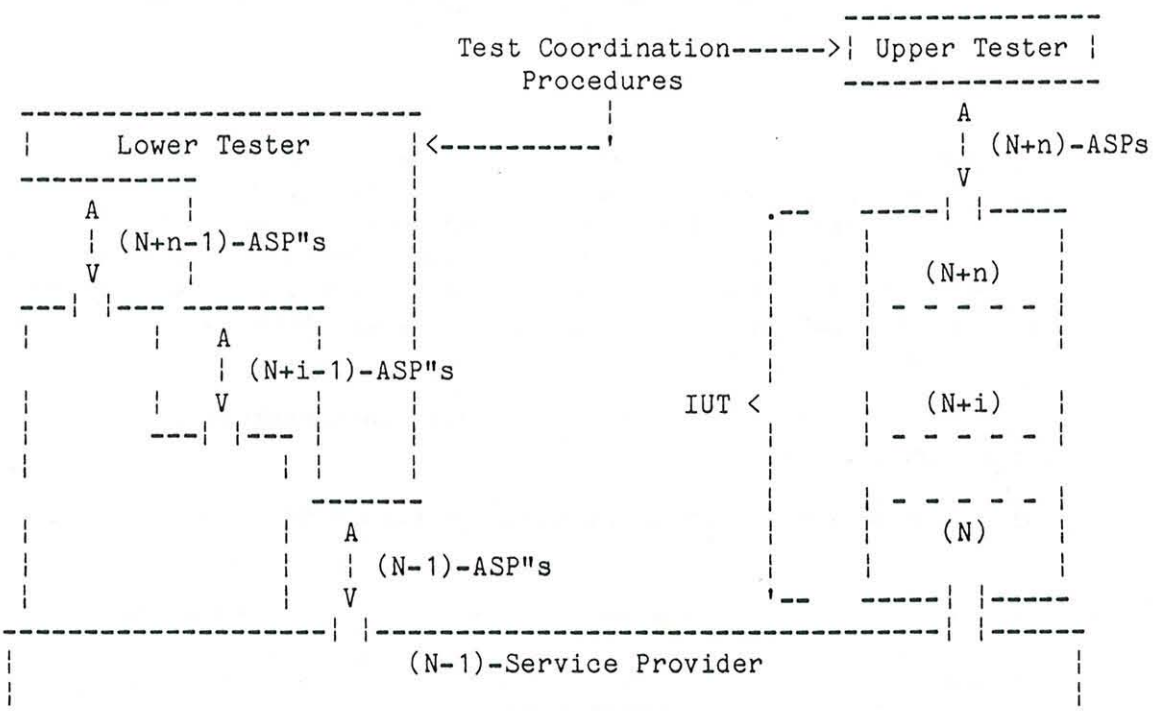


Figure 6. Incremental use of the DSE test method

Use of the DSE method on successive layers of a multi-layer IUT seems to be a more practical approach than using the DM test method to test the same IUT. Similarly, use of the RS method on successive layers of a multi-layer IUT would seem to be better than using the RM method. The main advantage of incremental embedded single-layer testing is that it requires less complicated test specifications.

5.8 Test Methods for Network Relay Systems

The abstract test methods discussed above all apply to testing protocol implementations in end-systems. Different methods are required for testing Network Relay Systems. Two have been proposed by SC21 and accepted by SC6/WG5. These are abstractions of the methods which have been used in practice at NPL [6]. They are as follows:-

(a) The case of testing a relay system from one subnetwork (loop-back testing).

This test configuration is shown in Figure 7.

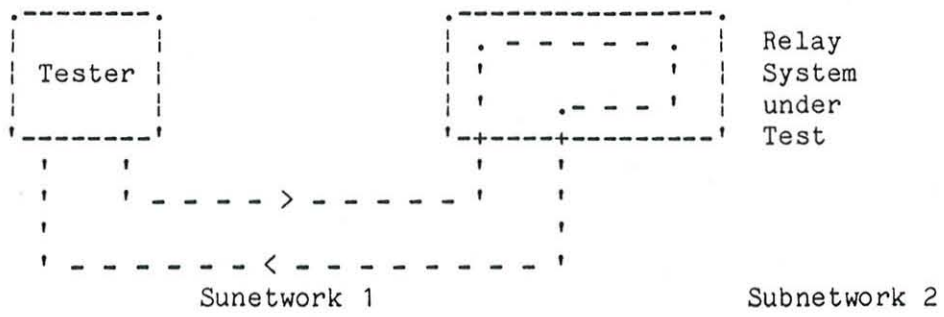


Figure 7. Loop-back Testing Configuration

For this testing configuration, one tester controls the test connections which are looped within the relay system. Therefore, it does not need two testing systems on different subnetworks and the problem of synchronization between the two testers is avoided. This testing configuration involves control and observation of PDUs on two connections on one side of the relay system, assuming that proper arrangements can be made for looping them together on the other side.

(b) The case of testing a relay system from two subnetworks (transverse testing).

This testing configuration is shown in Figure 8.

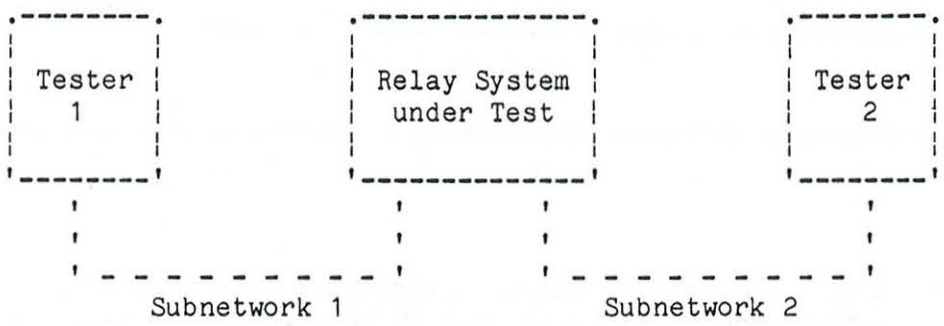


Figure 8. Transverse Testing Configuration

This testing configuration involves control and observation of PDUs on both sides of the relay system. If this test method is applied to testing the routing and relaying functions of the relay system, each of the testers may be required to simulate two or more end-systems.

Since Services below the Network Service are generally either unavailable or not well-defined, due to the fact that most protocols for real subnetworks were defined before Physical or Link Services were thought of, it is generally felt that the control and observation for Network layer testing needs to be interpreted in terms of PDUs rather than ASPs.

6. Applicability of the Test Methods

These abstract test methods are all applicable to the active testing of Transport, Session and Presentation layers without further comment. Their applicability to the Physical, Link, Network and Application layers requires further comment.

Further development may be needed for connectionless protocols and services.

6.1 Lower Layers

In the lower layers (Physical, Link and Network), the underlying service will not be end-to-end and so testing using the distributed and remote methods will need to be carried out over a single link rather than between end-systems. Furthermore, as with testing Network relay systems, the idea of specifying tests in terms of underlying service primitives is not likely to be of practical value; instead the tests will need to be defined in terms of PDUs. This is partly because the Link and Physical Services are not directly related to the protocols in those layers; partly because there is no service under the Physical layer; and partly because the protocols are technology dependent and in some cases do not correspond to the OSI layers.

The fact that these layers tend to be implemented in hardware or firmware makes it likely that the emphasis will be on testing multi-layer IUTs for these layers. The exception is the use of the LS test method to test the design of the algorithm for a protocol before it is put onto a chip.

6.2 Application Layer

With the application layer, there is no layer service above it which can be controlled or observed via (N)-ASPs at a SAP. However, for some, perhaps most, application protocols there is a defined service with service primitives some or all of which may be controllable or observable. If they are then the test methods can be applied by using these primitives rather than (N)-ASPs, but if they are not then only the remote test methods can be used. The extent to which control and observation of application service primitives will be possible will vary from one application protocol to another and may be different at the two sides because of inherent asymmetry in the application. For example, what can be controlled and observed for an initiator of a file transfer operation will be rather different from what can be controlled and observed for the file transfer responder (i.e. the filestore end).

Another issue occurs in the application layer, that of testing conformance requirements which go beyond the pure protocol exchanges. It is not only necessary to know that PDUs are exchanged in a valid manner but also that the exchanges have the desired effects in terms of the service provided to the user. This may require the definition of special application protocol dependent test methods to complement those described above. For example, it may be necessary to inspect the contents of a filestore by local means before and after certain file transfer tests.

7. Realization of Abstract Test Methods

In order to run the tests that will in due course be standardized it will be necessary to realize at least one of the abstract test methods. This will involve making mappings between the abstract concepts and real events and real systems. This section identifies some of the possible ways of realising lower testers, upper testers and test coordination procedures by reference to

approaches proposed and used by various research and development groups.

7.1 Lower Testers

The usual realisation of a lower tester for remote and distributed methods is known as an Active Tester. There are several different possible designs [7,8] not all of which give full control and observation of (N-1)-ASP"s. Some early designs were based on using a reference implementation of an (N)-entity, perhaps coupled with an exception generator [9] or configuration module [10] to give some flexibility in generating desired patterns of behaviour, particularly invalid ones. An alternative is to use an encoder/decoder module [10,11,12] to generate any desired sequence of PDUs, whether valid or invalid. However, to overcome the deficiencies of these approaches, a second generation design has been proposed [8,13] which will enable control and observation to be specified in detailed (N-1)-ASP" terms when necessary, or in higher level terms (e.g. (N)-ASP"s) when that is more appropriate. In addition, it may be necessary to use a portable testing unit, such as Cerbere [10,14], as part of the realisation of the lower tester when the underlying service is not end-to-end.

Whatever means is chosen for generating and recognising the (N-1)-ASP"s, an Active Tester will also need some form of Test Driver to control the operation of the test according to the executable test definition. Some Test Drivers are manual, some take commands from a file, and some use finite state based techniques [15]. Future ones may well be based on standard formal description techniques. Whatever approach is used, it is important that the executable test definition should be able to cover the whole tree of possible behaviours specified in the abstract test specification, not just a few of the main paths through that tree.

7.2 Upper Testers

Local and distributed test methods require the use of an upper tester. This observes and controls service primitives within the system under test. There must therefore be a mapping between abstract service primitives and real events in the system under test, but these need only be known to the implementor or supplier of that system.

The details of the realization of the upper tester can be entirely system specific, but it has been found advantageous to have a portable system independent approach to implementing upper tester functionality. The usual approach is called a Test Responder, but there are many possible designs [7] which vary in the demands they make on the system under test and in the amount of control and observation that they can perform. Another approach which has been recently proposed [16] is to implement a portable module called a Ferry in the system under test, the purpose of which is to relay all the (N)-ASP events over the network to a Test Driver in the same system as the Active Tester. This avoids having to design a special Test Responder with its inevitable compromise between portability and flexibility, but does require the system under test to support a module (the Ferry) which uses two connections simultaneously.

It has been proposed that there should be standardization of a single design of Test Responder but this has not yet been accepted. Instead the current focus of attention is on the possibility of standardizing the general functionality required by an upper tester.

7.3 Test Coordination Procedures

The major problem to be solved by the test coordination procedures is the synchronization of activities by the upper and lower testers. Research work has shown that this is a serious problem with the distributed test method. Loose synchronization by means of remote terminal access, use of the telephone and the setting to appropriate timer values has been shown to be inadequate, as might have been expected.

A better approach is to use a Test Driver-Responder Protocol (or Test Control Protocol) designed in conjunction with a Test Responder. There are two main design choices to be made for such a protocol. The first is whether to operate it on the same connection as the test traffic, or on a separate parallel connection going through the IUT, or on a separate parallel connection which interfaces to the Test Responder without going through the IUT. The second choice is whether to make each PDU of this protocol relate directly to one test event or to have PDUs which set up a test step involving several events up to the next synchronization point. All these approaches have some technical inadequacies and some add considerably to the complexity of the Test Responder.

An alternative approach is available if the Ferry idea is used. Since the control of the upper tester is then realised by a Test Driver in the same system as the Active Tester, synchronization between upper and lower tester becomes merely a matter of inter-process communication within a single system. The weakness is that there is a delay between activity by the Test Driver and corresponding activity by the Ferry.

Synchronization is, of course, much easier to solve with the local test method, since the test coordination procedures only need to operate within a single system. It may also be thought that the remote test method avoids the problem entirely. This is not so because, although there is no defined upper tester activity to be coordinated with the lower tester activity, experience shows that some undefined, system specific control of the system under test will be needed; and this will have to be synchronized with the lower tester activity, but the option of using a protocol to do it is almost certainly not available. So it is back to remote terminal access and use of the telephone.

8. Topics Requiring Further Work

The main topics which require further work for the standardization of OSI conformance testing are:-

(a) test notation;
(b) relationship between testing and FDTs;
(c) timing considerations;
(d) analysis and comparability of results;
(e) principles for use and design of multi-layer test suites;
(f) production of test suites.

It is desirable to have a single preferred test notation in which abstract test specifications can be written. Currently four candidates are under study in SC21 and a fifth has been used for the X.25 test suite by SC6. The SC6 notation is an informal one which is only applicable to simple tests using the RS test method. The ones being studied by SC21 are supposed to be able to handle complex tests for any test method. However, the tabular method (which is also proposed for Teletex tests by CCITT) is better at concentrating on the main paths rather than specifying the whole tree of behaviour for a complex test. The other three are an informal tree notation and the two ISO FDTs, Estelle and LOTOS. A comparison of the four is being made by using them to define some complex example tests [17] for the Session and Transport protocols.

Other aspects of the relationship between FDTs and testing are seen as having lower priority. They concern the suitability of each of the ISO FDTs for derivation of tests from formal descriptions of protocols and services. Current evidence [18] suggests that LOTOS is better suited for this than Estelle.

There are two main aspects of timing considerations: conformance requirements concerning timing and timers and how these should be tested; and the use of timers in testing for such things as the detection of inactivity. There are currently a great many questions on these aspects and very few answers [19].

Very little has been contributed so far on analysis and comparability of results. Yet a major reason for standardizing tests is to ensure comparability of results between different organisations which carry out the testing.

There is general agreement that the testing of multi-layer implementations is of major practical importance, but most of the work to date has gone into developing the single-layer testing ideas. It is currently unclear to what extend multi-layer test methods will be used in practice or how multi-layer test suites might be defined. It may well be that incremental embedded single-layer testing will be the dominant technique.

As part of the work of evaluating the conformance testing methodology and framework, it is proposed that trial test suites should be developed, probably for Transport, Session and an application protocol. These could then form the basis of the full test suites for these protocols. However, much effort is required to produce such test suites and it remains to be seen how quickly this will be forthcoming. At present the only test suite which is being developed by ISO is the one for X.25. CCITT is also working on a test suite for Teletex.

9. Conclusion

The ISO work on OSI conformance testing is now well advanced and has successfully generalised the work done by various research groups. A lot still remains to be done, but the energy and enthusiasm of the experts engaged in this work enables one to be optimistic about the outcome.

Acknowledgements

The author gratefully acknowledges the efforts of the ISO rapporteur group on OSI conformance testing, which he leads, without which this paper would not have been possible. He also wishes to acknowledge the support given to him by many members of the NPL Protocol Standards Group and the National Computing Centre COMMS-AID team. His own work at NPL has been supported by the Electronics and Avionics Requirements Board of the Department of Trade and Industry.

References

[1] ISO/TC97/SC21/WG16-1, Working draft for OSI conformance testing methodology and framework, ISO/TC97/SC21 N 410, Paris, February 1985.

[2] ISO/TC97/SC21/WG16-1, Draft answer to the question on conformance to OSI standards (WG1 Q 16 A), ISO/TC97/SC21 N407, Paris, February 1985.

[3] ISO/TC97/SC16/WG1, Checklist for protocol standards to permit conformance testing (revised), ISO/TC97/SC21 N 88, Copenhagen, June 1984.

[4] D. Rayner, Towards an objective understanding of conformance, in: Protocol Specification, Testing and Verification III, proc. 3rd international

workshop on this subject, held in Zurich on 31 May - 2 June 1983, edited by H. Rudin and C.H. West, North Holland, 1983, 477-492.

[5] R.F.L. Henley and D. Rayner, Implementation assessment of Transport and Network Services: an informal description of tests, NPL Technical Memorandum DNACS TM 5/81, July 1981.

[6] H.X. Zeng and D. Rayner, Gateway testing techniques, in: Protocol Specification, Testing and Verification IV, proc. 4th international workshop on this subject, held at Skytop, Pennsylvania, on 12-14 June 1984, edited by Y. Yemini, R. Strom and S. Yemini, North Holland, 1985, 637-655.

[7] G.W. Cowin, R.W.S. Hale and D. Rayner, Protocol product testing - some comparisons and lessons, in: Protocol Specification, Testing and Verification III, edited by H. Rudin and C.H. West, North Holland, 477-492.

[8] J.R. Pavel and D.J. Dwyer, Some experiences of testing protocol implementations, in: Protocol Specification, Testing and Verification IV, edited by Y. Yemini, R. Strom and S. Yemini, North Holland, 657-677.

[9] R.J. Linn and J.S. Nightingale, Some experience with testing tools for OSI protocol implementations, in: Protocol, Specification, Testing and Verification III, edited by H. Rudin and C.H. West, North Holland, 521-531.

[10] J-P Ansart, CERBERE, GENEPI, STQ: Three tools for protocol implementation testing, proc. 1st international workshop on Introduction to High Level Protocol Standards for Open Systems Interconnection, held in Paris, on 27-29 June 1983.

[11] D. Rayner, A system for testing protocol implementations, Computer Networks, 6, December 1982, 383-395.

[12] J-P. Ansart, GENEPI/A - a protocol independent system for testing protocol implementation, in: Protocol Specification , Testing and Verification II, proc. 2nd international workshop on this subject, held in Idyllwild, California, on 17-20 May 1982, edited by C. Sunshine, North Holland, 1982, 539-554.

[13] J.R. Pavel, A new approach to the design and construction of protocol testers, NPL Report DITC 54/85, April 1985.

[14] J-P. Ansart and J. Damidau, CERBERE, a tool to keep an eye on high level protocols, in Protocol Specification, Testing and Verification II, edited by C. Sunshine, North Holland, 529-538.

[15] P.W. Hobson (ed.), Implementation assessment of the OSI Network Service: the test definition language, NPL Report DITC 52/84, December 1984.

[16] H.X. Zeng and D. Rayner, The impact of the ferry concept on protocol testing, to be published in proc. 5th international workshop on Protocol Specification, Testing and Verification, held in Moissac, Toulouse, France, on 10-13 June 1985.

[17] ISO/TC97/SC21/WG16-1, Test examples, ISO/TC97/SC21 N 412, Paris, February 1985.

[18] ISO/TC97/SC21/WG16-1, Conformance testing and FDTs, ISO/TC97/SC21 N 411, Paris, February 1985.

[19] ISO/TC97/SC21/WG16-1, Time dependencies and timers, ISO/TC97/SC21 N 413, Paris, February 1985.

DISCUSSION

While mentioning the static and dynamic aspects of conformance tests, the speaker, in response to Professor Randell's question on the static aspects, explained that those aspects stand for the capabilities of a protocol being tested. As he went on to explain the requirements for conformance tests, Professor Whitfield asked what the speaker meant by conditional requirements. The speaker answered that when certain (boolean) conditions become true, those requirements will become mandatory; otherwise they are unspecified. As he explained the optional requirements as the ones that can be set at will, Professor Randell pointed out that if all the requirements were mandatory, PICS (Protocol Implementation Conformance Statement) would not be necessary.

During the post-talk discussions, Professor Randell questioned how the test protocols can be standardised when the protocol specifications themselves are inadequate and asked about the merits of public demonstrations of system linking. The speaker stated that a great deal of work has to be done in linking demonstrations; he remembered one such demonstration in Computer Conference at Las Vegas participated by about fourteen vendors. However, the results of the demonstration were not very fruitful, in his opinion. He suggested that standard test procedures be conducted at various test centres to achieve a level of commonality instead.

Finally, Dr. Cerf raised a question whether it is possible to have a transportable test protocol, especially for X25 systems that are getting increasingly popular. The speaker replied that one has been developed in France. He also pointed out that those transportable test-protocols tend to have a poor performance due to its flexibility.