PAPER Special Issue on Broadband ISDN —Application, Networking and Management—

Applying OSI Systems Management Standards to Remotely Controlled Virtual Path Testing in ATM Networks

Satoru OHTA† and Nobuo FUJII†, Members

SUMMARY Asynchronous Transfer Mode (ATM) is an information transport technique that well supports Broadband ISDN (B-ISDN). One unsolved problem to the perfection of ATM networks is to provide a testing environment that conforms to some standardized network management scheme. From this point of view, remotely controlled virtual path testing is considered in this paper. Remotely controlled virtual path testing should be executed through the standardized Telecommunications Management Network (TMN) model, which employs the OSI systems management concept as the basis of information exchange. Thus, this paper addresses the two issues that arise when OSI systems management standards are applied to virtual path testing. One issue is to define relevant information models. The other issue is to provide test resources with a concurrency control mechanism that guarantees a consistent test environment without causing deadlocks. To resolve these issues, technical requirements are clarified for the remote control of test resources. Next, alternatives to the concurrency control mechanism are shown and compared through computer simulations. A method of defining information models is then proposed. The proposed method ensures the easy storage and retrieval of intermediate test results as well as permitting the effective provision of concurrency control for test resources. An application scenario is also derived. The scenario shows that tests can be executed by using standardized communication services. These results confirm that virtual path testing can be successfully achieved in conformance with the OSI systems management standards.

key words: *communication systems and transmission equipment, ATM, testing, network management, OSI systems management, managed object*

1. Introduction

Asynchronous Transfer Mode (ATM) is an information transport technique specially designed to support Broadband ISDN (B-ISDN).[1] Because ATM is relatively new, many studies have been performed to resolve the various problems associated with ATM.[2]–[12] One remaining problem is to establish a network management system for ATM-based networks. Network management includes areas such as fault or performance management,[13] in which various tests on network resources play an important role in obtaining information about faulty or degraded resources. Testing, therefore, is a significant subject in constructing ATM-based networks.

Manuscript received August 28, 1992.
Manuscript revised November 5, 1992.
† The authors are with NTT Transmission Systems Laboratories, Yokosuka-shi, 238-03 Japan.

This paper deals with the testing of virtual paths. The definition and virtues of virtual paths can be found in the literature.[3]–[7] Current studies on virtual path testing address several points of view. For example, some reports have presented the principles of performance measurement of virtual path/channel connections for the purpose of testing and monitoring.[8]–[11] Moreover, the performance parameters examined by testing are now being discussed in standardization organizations. Several experimental ATM systems have employed testing mechanisms for verifying virtual path/channel connections (for example, see Ref. (12)). However, no studies has yet to propose methods for remotely controlling virtual path tests.

In remote testing, the network resources must be remotely controlled using the Telecommunications Management Network[13] and a standardized interface. Across this interface, information is exchanged in conformance with the OSI systems management[14] standards which specify information exchange between communicating processes for the purpose of management. Here, one process is the manager, while the other is the agent that manages managed objects representing network resources. To apply the standards to a particular management problem, every associated information model such as managed objects must be defined, and the control of resources must be achieved through standardized communication services/protocols. Additionally, we note that a very recent contribution appears to introduce transaction processing capabilities into the OSI systems management protocols.[22]

The OSI systems management standards have been specified generi cally for a wide range of communications management problems. For testing functions, a draft standard has been recently proposed[15] which provides the fundamental concept for defining information models. Nevertheless, no information model definitions specific to virtual path testing have been proposed so far.

The information models need to be defined so as to include a concurrency control mechanism for the consistent execution of tests. Reference (16) has described a method that implements concurrency control in a manner conforming to OSI systems management

standards. However, there exist several alternative concurrency control schemes other than the method of Ref. (16). Thus, these alternatives must be compared, and the scheme that most appropriately defines the information models must be determined.

This paper addresses the two issues described above: the comparison of alternative concurrency control schemes and the information models specific to virtual path testing. The former issue is investigated by considering the stability of control actions and evaluating scheme performance by computer simulations based on a model representing a virtual path network. The result is that the scheme utilizing the wound-wait algorithm is to be preferred. As for the latter issue, a method of defining the information model is proposed. The proposed method enables the easy storage and retrieval of intermediate test results together with the effective provision of concurrency control. These benefits are brought about by defining information models that correspond to the functions performed by a test resource. Moreover, this paper presents an application scenario, in which resources are controlled through standardized communication services. These results confirm that the remote testing of virtual paths can be successfully achieved in conformance with the OSI systems management standards.

This paper first clarifies the technical requirements of a control scheme for virtual path testing in Sect. 2. Section 3 explains international standards applied to virtual path testing and the assumptions used in this paper. In Sect. 4, alternative concurrency control schemes are identified and compared. Finally, in Sect. 5, ways of defining relevant information models and application scenarios are deduced from the requirements.

2. Requirements for Virtual Path Testing

Virtual path testing can be initiated after observing some degradation in performance, as a regular maintenance action, or as a check before bringing into service. Testing must be able to distinguish the kind of degradation and find the exact cause. For this purpose, it may need to examine performance parameters such as connectivity, bit error ratio, aborted cell ratio, errored cell ratio, and incorrectly inserted cell ratio. Several measurement schemes for obtaining these parameters have been reported[8]–[11] to test or monitor virtual path/channel connections.

In the schemes proposed to date, virtual path performance must be measured either in service or out of service.[8],[9] Out-of-service schemes offer accurate performance measurement[8] by loading the path with arbitrarily specified test traffic while the service provided by the path is interrupted during testing. Such schemes are useful to examine connectivity before a service is provided. In out-of-service schemes, user

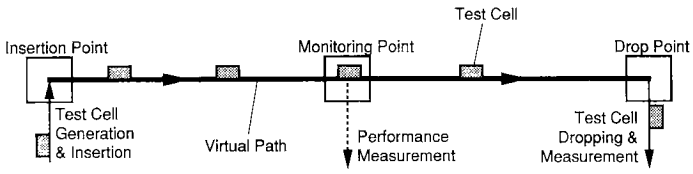


Fig. 1 The principle of virtual path testing.

cells need to be dropped at the insertion point to maintain the specified traffic load, and VP-AIS (AIS: Alarm Indication Signal) cells are inserted to notify the non-availability of the path. These operations are not necessary in the in-service scheme which does not necessarily disturb service.

In either type of measurement scheme, the performance is measured by using test cells of a specified format. These cells are OAM cells labeled with the same VPI as the virtual path to be tested. The cells can also carry other information used for counting the number of bit errors, aborted cells, or misinserted cells. The test cells are injected into the virtual path to be tested at the specified insertion point. The inserted cells leave the path at the specified drop point, and path performance is evaluated through the test cell information. The types of cells examined at the drop point differ with the measurement scheme. Information cells arriving between test cells are counted for the in-service measurement of aborted/inserted cells, while only test cells are examined in out-of-service testing. In either case, the drop point must have the functions needed to summarize the number of bit errors, errored cells, aborted cells, and inserted cells. To localize failures effectively, such summarization functions may be placed at points between the insertion and drop points. These intermediate points are referred to as monitoring points. At a monitoring point, test cells are not dropped but measurements are executed in the same way as the drop point. This measurement principle is summarized in Fig. 1.

This paper deals with the problem of executing the testing scheme described above through remote control. For this purpose, it is necessary to identify the requirements for remotely controlling the network resources involved in testing. Some of these requirements originate from characteristics specific to ATM or B-ISDN. One significant characteristic is that the additional traffic load imposed by test cells can affect network performance. Thus, the test traffic must be specified so as not to induce congestion which would degrade network performance. Note that some B-ISDN services such as video applications, are basically asymmetrical. For the paths transferring such services, a loop back test[10] may not be applicable because required test traffic load can not be supported in both directions. This means that one-way testing is mandatory for virtual paths, though a loopback or echo back test may be useful for simple connectivity checks.

Another issue arises from the performance parameters specific to ATM. Although the aborted cell ratio and inserted cell ratio are important performance parameters, cells are very seldom aborted or inserted. Therefore, measurement may need to be performed over an extremely long time to obtain accurate values of the aborted cell ratio or inserted cell ratio. For example, if the cell loss probability is 10^{-7} for the path to be tested, it will take about one hour on average, to observe 10 aborted cells if the test traffic rate is 10 Mb/s.

Additional requirements are derived from the general characteristics of transmission paths. First, test signals must not be received by the user. Furthermore, it must be possible to guarantee the injection of test signals during testing. To satisfy these conditions, control of network resources must be done in a specified order. Finally, the resources associated with a test are often shared among a number of paths. This necessitates concurrency control which is discussed in detail later.

By observing the above considerations, the following requirements should be considered so as to apply OSI systems management to virtual path testing.

- One-way testing must be provided. This leads to a testing environment in which two or more manager-agent models are involved (see Sect. 3).
- Long term measurements must be supported. This demands several test control functions, such as suspension, resumption, monitoring, or forcible termination.
- Resources associated with a test should be set in a consistent order to guarantee the injection of test signals and prevent user receipt.
- Concurrency control is needed because test resources are shared.

3. International Standards

CCITT has recommended the Telecommunications Management Network (TMN) concept as the basis of telecommunication network management.[13]

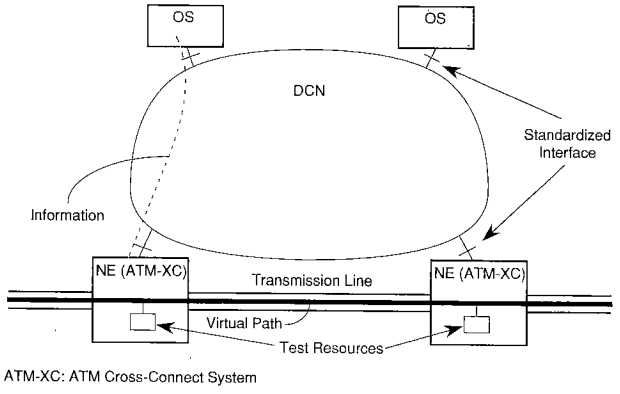


Fig. 2 The TMN model applied to an ATM network.

Figure 2 illustrates how the TMN concept accommodates an ATM network that supports virtual path testing. In this figure, the Operations System (OS) performs various network management processes by remotely monitoring or controlling the Network Elements (NEs) of the ATM network. Resources associated with virtual path testing are included in the Network Elements. The NEs and the OS are connected to the Data Communications Network (DCN) which transports information between the OS and the NEs. Standardized interfaces link the OS and the NEs to the DCN. Information is exchanged across these interfaces according to the OSI systems management concept. This means that the information models of remote virtual path testing must be defined in conformance with OSI systems management standards.

Figure 3 shows the OSI systems management model[14] as applied to the TMN model shown in Fig. 2. Suppose that the OS is communicating with an NE to perform a network management operation. The OS corresponds to a managing open system, and the NE represents a managed open system in the context of OSI systems management. Here, the process in the managing system takes a manager role, while that in the managed system takes an agent role. Besides these systems, the network resources being controlled or monitored are represented as the managed objects that are managed by the agent. The managing system can remotely control or monitor the resources by sending operations to or receiving notifications from the managed objects.

Reference (15) specifies the test management function in the OSI systems management model. In this specification, the manager process that issues test operations is called the test conductor, while the agent process receiving test operations is termed the test performer. Test operations issued by the test conductor and directed to a managed object must comply with the test action request receiver (TARR) functionality. The types of tests supported by the test performer are described as "test categories." A test is defined as either an asynchronous test or a synchronous test. The former can be monitored and controlled, while the latter can not. For an asynchronous test, a Test Object is defined to hold information pertaining to the test. A test invocation is a specific test instance, while a test session is defined as a set of test invocations. In this paper, a test session means the set of all test invocations

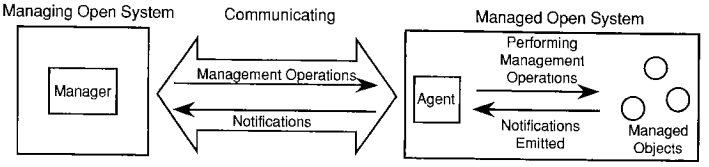


Fig. 3 The OSI systems management model.

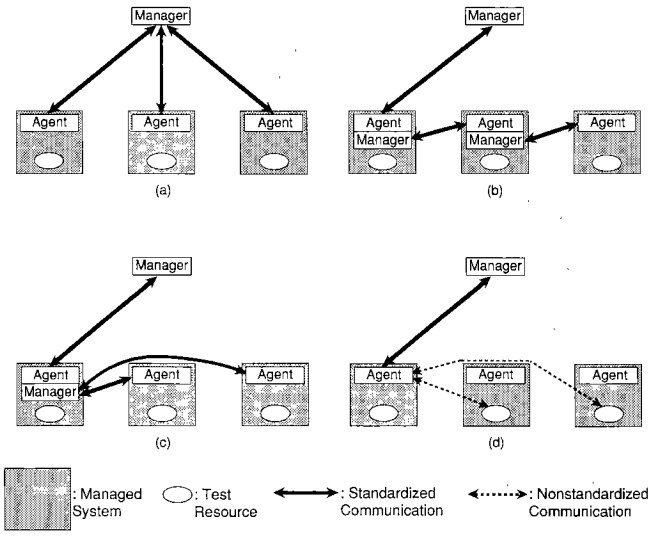


Fig. 4 Alternative ways of applying manager-agent relations.

concurrently issued to the cell drop point and monitoring points to examine a single virtual path.

In one-way virtual path testing, the cell insertion point is geographically separated from the cell drop or monitoring points. Therefore, it is natural to assume that the insertion point is always managed by a managed system different from those managing the drop and monitoring points. In addition, the managing system invoking a test is assumed to know the following before test invocation:

- traffic load of test cells,
- destination addresses of managed systems including insertion, monitoring, and drop points,
- identifiers of links or equipment to be accessed, and
- VPIs (virtual path identifiers) of the virtual path to be tested.

As described above, two or more managed systems are concerned with virtual path testing. In this environment, there exist several alternative ways of applying manager-agent relations. These alternatives are categorized as follows.

(1) Manager-agent relations are applied in parallel. That is, the test conductor exchanges information directly with each managed system as shown in Fig. 4 (a).

(2) Manager-agent relations are applied recursively. The test conductor exchanges information with only one managed system. This system generates another manager-agent model, and so on. Figure 4 (b) illustrates this scheme.

(3) The test conductor exchanges information with only one managed system. This system exchanges information with all other managed systems as shown in Fig. 4(c).

(4) One agent manages all resources through nonstandard communication paths as depicted in Fig. 4 (d). Namely, only one manager-agent relation is employed.

Testing can be implemented only after choosing one of these alternatives. The system processing load may be a criterion for such a choice. From this viewpoint, alternative (1) has the characteristic that the load is heavier on the managing system than on the managed systems. On the other hand, one test performer is more heavily loaded than the others in alternatives (3) or (4). In alternative (2), there is no particular system on which processing load is concentrated. Thus, if test performer load is to be decreased, then alternative (1) should be chosen. Conversely, another alternative, (2)-(4), must be selected if test conductor loads are to be decreased.

Although the above criteria will help in comparing the alternatives, it is still difficult to determine the best choice at present. This is because we have no information as to which system is most critical in terms of processing load. Furthermore, it is not known whether virtual path testing generates serious processing loads on systems or not. If testing does not impose any heavy load, the alternatives may yield the same overall performance. Therefore, this paper concludes that further studies are needed before the best alternative can be determined. Nevertheless, it is necessary to decide a method of applying manager-agent models to define information models. For this reason, only alternative (1) shown in Fig. 4(a) is considered hereafter. Note that this selection should not be seen as indicating its superiority.

4. Concurrency Control

Concurrency control is one of the issues to be addressed in applying the OSI systems management standards to virtual path testing. Concurrency control is indispensable for the consistent execution of tests. Also, the definition of managed objects and the application scenario are affected by the mechanism used to implement concurrency control. The problem is described as follows.

In general, a resource involved in testing is shared by the virtual paths multiplexed on the link or that pass through common equipment such as a cross-connect. Thus, two or more test performers may initiate test sessions using the same test resource concurrently. This leads to inconsistent setting of test environments and faulty test results. Therefore, while testing a virtual path, the test environment must be kept fixed to obtain consistent measurement results. The environment, once set, should not be disturbed by any other test session. This implies that every object associated with the testing environment needs to have a "write lock" mechanism, which prohibits any other test session from resetting its attributes. Furthermore, since there are two or more associated resources and there

may be two or more test sessions, a test session may become deadlocked. Figure 5 shows a simple example of deadlock. In this figure, test session A can not start because it can not lock managed object # 2 which has already been locked by test session B. Similarly, since test session A has locked managed object # 1, test session B can not run. As a result, both test sessions wait permanently. Thus, managed objects must have a concurrency control mechanism that supports "write locking" as well as a deadlock prevention method.

The concurrency control mechanism selected should correctly work among distributed objects managed by different agents in the model shown in Fig. 4(a). Additionally, the mechanism needs to be implemented according to the OSI systems management standards. That is, it must be described by using managed objects, an application scenario of the managing system, and standardized communication services.

Fortunately, these requirements are satisfied by employing the method proposed in Ref. (16). This method defines a concurrency control package that is included in all managed objects related to the test environment. The package features the use of transaction IDs. The write lock mechanism is achieved by applying the CMIS filtering function to the transaction IDs. The term "transaction" corresponds to a test session in the context of this paper. A transaction ID contains the time of transaction invocation, that is, a timestamp. To avoid deadlocks, the method of Ref. (16) exploits the "wound-wait" algorithm,[17] which implemented as follows. When a lock request from transaction T_i conflicts with another request from transaction T_j at a managed object, the timestamps of the requests are compared. If T_j is younger than T_i, then the object emits a "wound" notification to the manager executing T_j. Otherwise, T_i waits for the object to be unlocked by an unlock operation from the manager executing T_j. These steps are described in the behaviour of the concurrency control package. A manager receiving a "wound" notification aborts its transaction if it is in the locking phase. In this way, wound-wait can be incorporated into the OSI systems management environment. The scheme is summarized in Fig. 6.

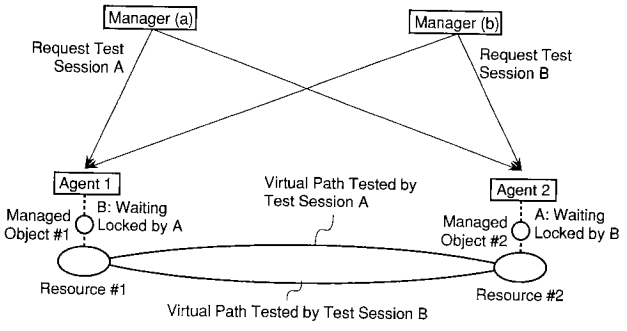


Fig. 5 An example of deadlocked test sessions.

By observing Fig. 6 carefully, it can be seen that other deadlock prevention algorithms can also be used with slight modifications. Figure 7 shows two such alternatives: Fig. 7 (a) shows the "running priority" algorithm[18] and the "wait-die" algorithm[17] is shown in Fig. 7 (b). The running priority algorithm emits a wound notification whenever a conflict is generated. However, running test sessions can never be aborted and restarted by just the wound notifications; only waiting test sessions are aborted and restarted. It is easy to see that the scheme of Fig. 7(b) follows the definition of the wait-die method. Other deadlock prevention techniques may also be applicable. The resource preordering technique (for example, see Ref. (19)) can be achieved by locking every object in a specified order, say, the lexicographic order of object identifiers. Timeout[20] is another feasible alternative.

The important issue is to choose the best deadlock avoidance strategy. Among the alternatives described above, resource preordering has the disadvantage of increased response delay.[19] The timeout technique is also weak in that it is difficult to find an appropriate timeout period.[20] References (18) and (20) use computer simulations to compare all or some of the three remaining alternatives. However, no definitive choice

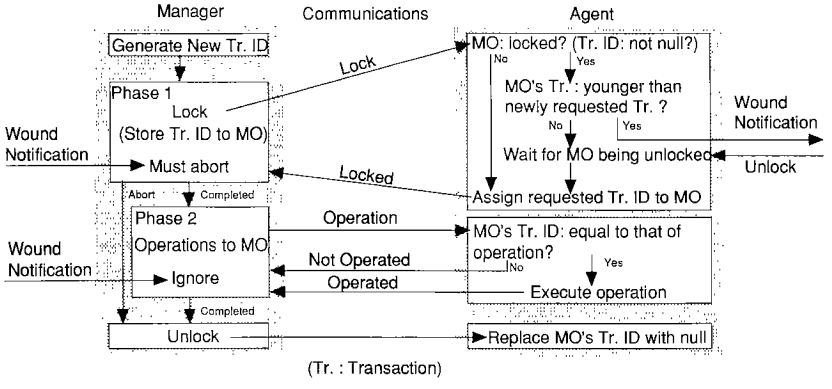


Fig. 6 The concurrency control scheme of Ref. (16).

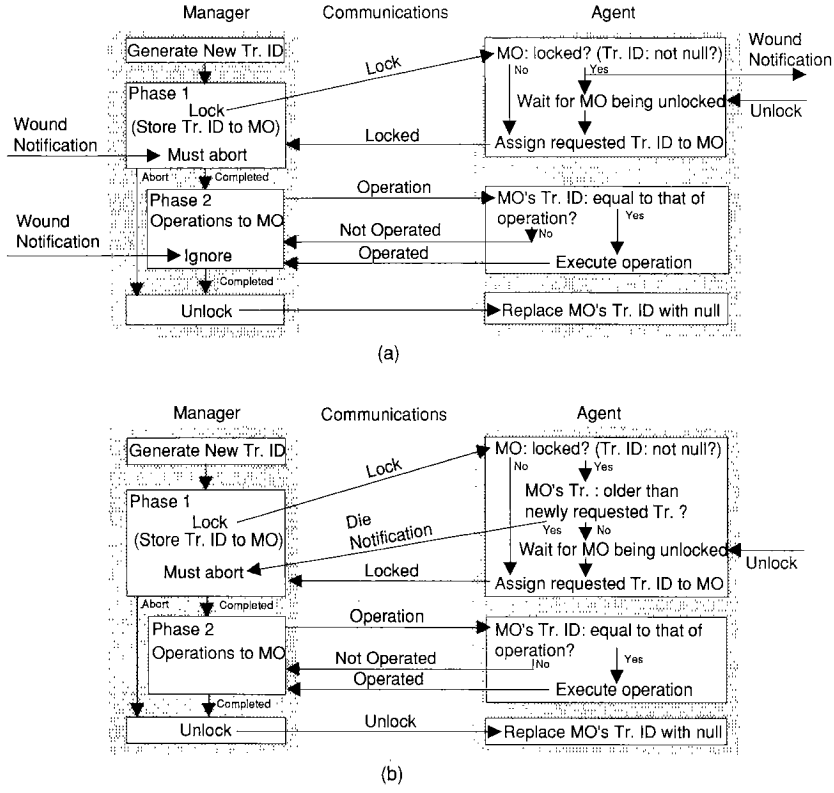


Fig. 7 Alternatives for concurrency control.
(a) running-priority. (b) wait-die.

can be made on the basis of these references. This is because some results are contradictory due to differences in the models examined.[20] In addition, since the references basically considered database systems, the simulation models were not necessarily appropriate for virtual path testing. Therefore, this study has performed more accurate computer simulations using a model specific to virtual path testing.

The employed model represents a virtual path network consisting of 5 nodes; a relatively small network. In the network, m virtual paths are set between every pair of nodes. The insertion point of test cells is the source node of the path to be tested, while the drop point is the destination node. Each node has n cell insertion resources and n cell drop resources. Each resource is represented by a managed object. Before a test is executed, it is necessary to lock the managed objects representing the cell insertion resources and the cell drop resources. For each virtual path, a test request is generated by the managing system X seconds after the previous test is completed. Variable X is randomly distributed using an exponential distribution with mean value x. The execution of the requested test starts as soon as both managed objects representing the insertion and drop resources are locked. The test execution lasts for Y seconds, and the managed objects are immediately unlocked at the completion of execution. Variable Y is exponentially distributed and its mean value is y. It takes Z_1 seconds for a managed object to receive a lock request after a test request is generated. If an agent generates a wound or die notification, the associated transaction is aborted by either agent of the end nodes of the path Z_2 seconds later. An aborted transaction is restarted Z_3 seconds after the aborting process is completed at both agents of the end nodes. Variable $Z_i (i=1, 2, 3)$ is randomly determined according to the normal distribution of mean value z_i and standard deviation s_i.

During the simulation, we recorded the average delay and average restart number. The average delay is defined as the averaged value of time taken from request generation to the start of test execution. The average restart number is the average number of restarts taken by a test request from generation to the start of test execution. These values were measured by varying mean test execution period y with constant values of m, n, x, z_i, and s_i. The load on a test resource is heavier for larger values of y. The measurements were made for 3000 test requests for each value of y.

The parameter values were determined in the following way. In a real network, X is expected to be a variable ranging from several days to years. Thus, the simulation assumed its mean value x to be 6×10^5 seconds ($\approx$1 week). Constant $z_i (i=1, 2, 3)$ is determined by delay in message communication as well as processing but is expected to be no more than a few

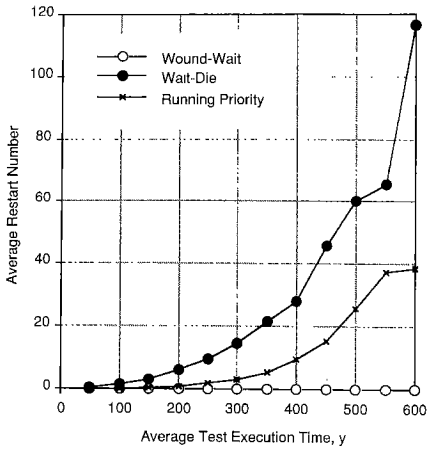


Fig. 8 Average restart number versus mean test execution period for alternative concurrency control schemes.

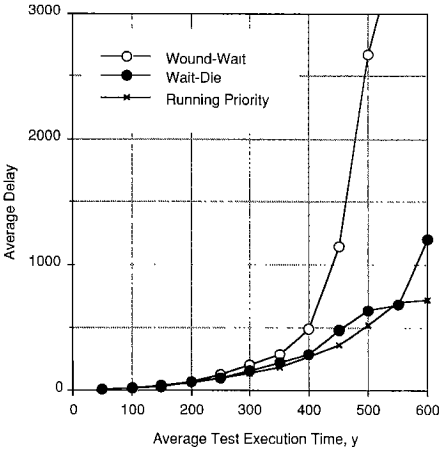


Fig. 9 The average delay versus mean test execution period for alternative concurrency control schemes.

seconds. Therefore, z_i and s_i were set 3 seconds and 0.9 seconds. Variable Y may vary greatly; for example, several seconds for a simple continuity check to one hour for cell loss measurement as described in Sect. 2. Presently, it is very difficult to estimate the statistical characteristic of variable Y. For this reason, its mean value y was varied over such a range that difference between the alternative methods was made clear. Integers m and n were assumed to be 100 and 1. The employed values for m and n are not critical because these parameters depend on the real network size and equipment implementation issues.

Figure 8 shows the average restart number versus mean test execution period y. As shown in Fig. 8, the average restart number is much less for the wound-wait algorithm than for the other two methods. Since frequent restarts lead to heavier loads on the DCN, OS, and NEs, the wound-wait algorithm is superior in this regard.

The average delay is plotted against y in Fig. 9. Since testing is not a strict real time operation, a delay of several minutes is considered to be allowed. Thus, the figure shows that all methods satisfy the requirement for $y \leqq 350$. For $y > 350$, wound-wait induces larger delay than others and thus may not be applicable. However, other two methods are also not applicable because of their excessive number of restarts. To see this, let us estimate the number of messages that a manager must process assuming that the manager handles 1000 virtual paths. Suppose that wait-die is employed and $y = 600$. It is not difficult to see that the manager must send or receive at least 5 messages per restart. From this observation and the employed parameter values as well as the simulation result, it is derived that the manager has to handle about 1 message per second for restarting tests. This imposes a significant load on DCNs and the managing system. Therefore, no methods are appropriate for $y > 350$. Nevertheless, the delay for wound-wait can be improved by decreasing the number of virtual paths per test resource. That is, wound-wait is applicable for larger y if a sufficient number of test resources is provided.

In addition to delay and restart number, stability is also useful for ranking the alternatives. From this viewpoint, both wait-die and running priority have drawbacks. With wait-die, a transaction may be restarted repeatedly,[20] and the older transaction tends to slow down.[17] The running priority scheme can also repeatedly restart a transaction. Moreover, this scheme does not guarantee that transaction will finish sefely.[20] These characteristics are not acceptable in network management. By contrast, wound-wait has good stability. With wound-wait, a restarted transaction is not restarted by the same conflict, and the oldest transaction is assured of finishing.[20] This consideration and the simulation results imply that wound-wait is the best method for virtual path testing.

5. Defining Information Models

The remote testing of virtual paths makes it necessary to define information models such as managed objects. The models are fundamental to realize communication between a manager and an agent. An application scenario must be also clarified to permit test execution. The models and scenario should follow international standards as well as satisfy all the requirements stated above. This section derives the relevant definitions of information models and scenario to meet the requirements specific for virtual path testing and the restrictions demanded by the relevant standards.

The international standard for test management function requires the definition of test categories associated with virtual path testing. One issue is to determine whether the test is asynchronous or synchronous. Only asynchronous tests need be considered

because monitoring during test invocation and control operations such as suspending/resuming are necessary to flexibly accomplish long term tests. Obviously, test categories need to include suspending and resuming operations. For the test conductor to set resources in a consistent order, a test should be terminated upon the receipt of a request from the test conductor. Therefore, the test termination type must be defined as a "solicited" termination.

Since testing is asynchronous, it is necessary to appropriately define a Test Object. A Test Object is defined as an object that holds information pertaining to the test.[15] In the case of virtual path testing, it is useful to define a Test Object such that its attributes represent data measured by the test resource. This enables a test conductor to retrieve information about measurement values or the progress of the measurement process by using a "get attribute value" operation. It also makes it easy to record intermediate test results in a Test Object when a test is suspended.

A Test Object can be defined so as to represent a test resource. However, a Test Object is created at test invocation[15] and does not necessarily correspond to a resource in a one-to-one manner. To recognize this, consider the case that a test is suspended and the associated test resource is used by a newly invoked test. For this case, there exist two Test Objects associated with one test resource. Thus, a Test Object is related to a test invocation more tightly than to the resource. Due to this characteristic, it is difficult to provide comprehensive concurrency control for Test Objects. This is because a "lock" operation on a Test Object does not necessarily mean that the associated resource is locked if two Test Objects are associated with the same resource; a "replace value" operation on the other object may change the setting of the resource.

Test Objects must be defined at every drop or monitoring point. No Test Object exists at the insertion point because no measurement is executed and no test results are obtained. Considering the above discussion, two distinct functions of a test resource at a drop or monitoring point are identified for the effective definition of the Test Object. These functions are termed the connecting function and the summarization function. The connecting function supports the selection of the cells to be examined, execution of cell dropping/monitoring, and the insertion of AIS cells for out-of-service testing. For this purpose, the function stores the VPI of the tested path, the point to be accessed, and a monitoring/dropping indicator. Namely, the connecting function determines the test environment. The summarization function observes and calculates the numbers of bit errors, aborted cells, and inserted cells using the cells selected by the connecting function. The Test Object is defined as the summarization function of test resources at drop and monitoring points. The connecting function is expressed as another managed object; which is referred to as an Associated Object in the standard testing context.[15] This Associated Object contains the concurrency control package mentioned in Sect. 4. These functional object definitions are reasonable considering the following points.

- Since the Test Object holds data obtained by the test, it is easy to retrieve measurement information or maintain intermediate test results while the test is suspended.
- The Test Object does not include any attribute that needs to be "write locked." The Test Object attributes are essentially read-only because they are data updated during the measurement process. Thus, they do not need a concurrency control mechanism.
- The attributes requiring a concurrency control mechanism are all contained in the Associated Object, which is defined as to correspond to the resource in a one-to-one manner. Thus, a consistent test environment is assured.

To achieve the definition described above, the Associated Object must have attributes representing the VPI of the virtual path to be tested, the test mode, and the access mode. The test mode attribute indicates if the test is executed in-service or out-of-service. The access mode attribute shows whether the test resource is a drop point or a monitoring point. The Associated Object behaviour is such that the test resource extracts the cells of the tested virtual path from all received cells according to the VPI and test mode attribute values. In addition, the resource determines whether to drop the selected cells or not according to the access mode attribute value. Moreover, the object must include the concurrency control package performing the wound-wait algorithm. The GDMO description of this package is seen in Ref. (16).

The Test Object, on the other hand, needs to have attributes representing values of performance parameters measured from selected cells by executing measurement algorithms implemented in the test resource. There needs to be another attribute to store the specified measurement time length. The behaviour is such that the Test Object reads periodically the counters of the test resource and stores the values into the performance parameter attributes. This procedure continues for the time specified in the measurement time attribute. If a test is suspended, the measurement time attribute is updated to the remaining time. If the suspended test is resumed, the performance parameter attributes are updated by the resource counter values for the time indicated in the measurement time attribute. Note that measurement can only be started after confirming the establishment of the test environment. Therefore, measurement must be initiated by a request from the test performer. This requires the Test Object to have an action that initiates measurement.

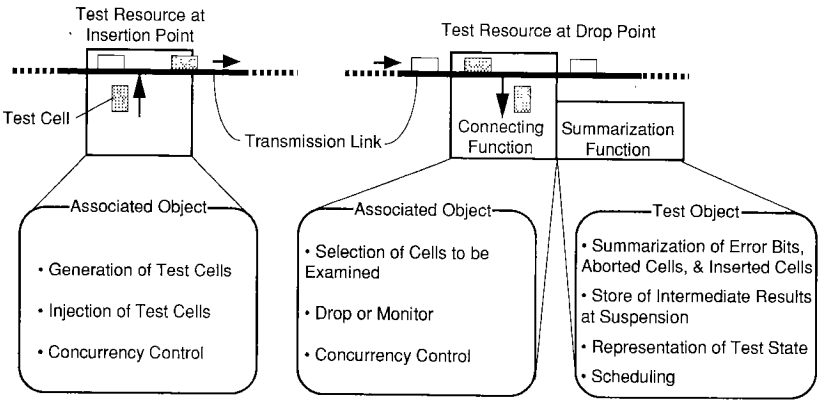


Fig. 10 Test resource functions and object definitions.

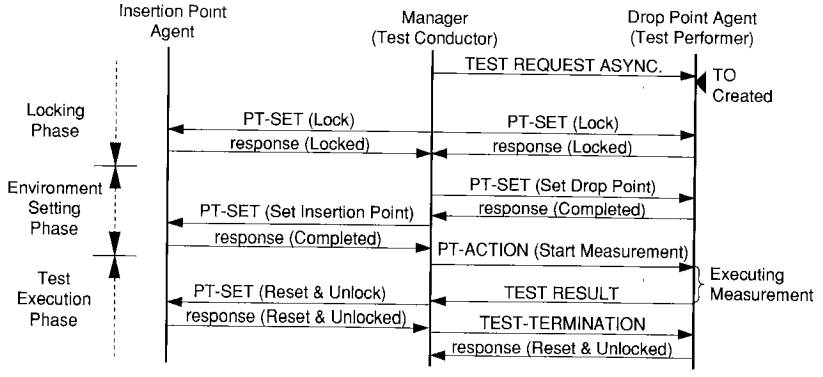


Fig. 11 An example of an application scenario (1).

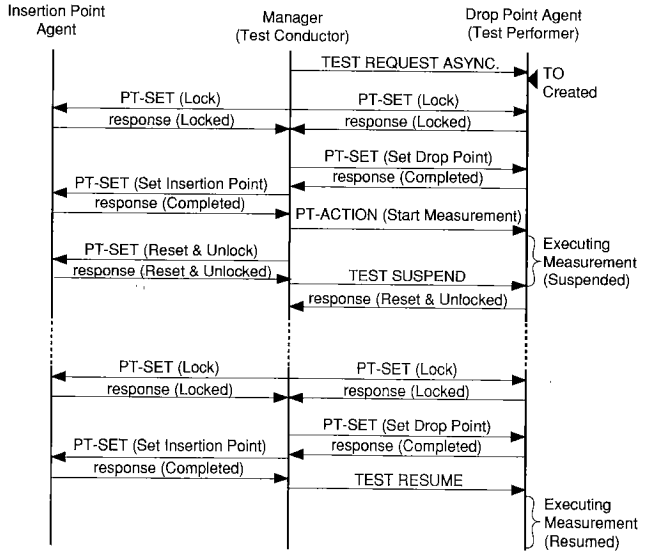


Fig. 12 An example of an application scenario (2).

Another Associated Object is defined to represent the resource at the insertion point. This object has the attributes needed to specify the VPI, traffic condition of test cells, and the test mode. The object has a behaviour such that its corresponding resource inserts test cells according to the condition indicated by the attributes. The concurrency control package is included also in this object to hold the attributes to the specified values during the test. The definition of the concurrency control package is the same as the one used in the Associated Object defined for a drop or monitoring point. The above manner of defining

objects is summarized in Fig. 10.

To perform a test, an application scenario should be described as well as the information models. The scenario needs to incorporate functions such as initialization, termination, suspension, and resumption. The scenario must be defined so as to set the resources in a consistent order and accomplish the concurrency control mechanism. The scenario must utilize standardized communication services between the manager and agents. Figure 11 illustrates a scenario involving the test initialization function and the test termination function.

From Fig. 11, it is observed that the scenario consists of the locking phase, the environment setting phase, and the test execution phase. In the locking phase, the manager requests the locking of the managed objects associated with the test by using the PT-SET service.[21] This is done before other operations to achieve the concurrency control mechanism described in Fig. 6. The test environment is established after confirming that all test resources have been locked by the test sessions requested. During the environment setting phase, the manager requests the injection, monitoring, and dropping of cells by using the PT-SET service. This is executed before test execution to assure the injection of test cells during the measurement period. The injection of cells is requested after the completion of drop point setting was confirmed to ensure that test cells are not sent to the user. Measurement is started by the request of the manager through the PT-ACTION service,[21] and the result is reported through the RESULT REPORT service.[15] The resource at the drop point must be reset after confirming the termination of cell injection. Thus, the test is terminated by the request from the manager using the TEST TERMINATION service[15] after receiving the response that confirms the resetting of the drop point. Similarly, the suspension and resumption processes are shown in Fig. 12.

6. Conclusion

This paper has addressed the application of OSI systems management standards to remotely controlled virtual path testing. We have clarified technical requirements to determine necessary information models and scenarios. Among these requirements, concurrency control was identified as an important issue. Therefore, this paper examined several alternative concurrency control methods. These alternatives were compared through computer simulations and a consideration of control action stability. As a result, the scheme of Ref. (16), which employs the wound-wait algorithm, was determined to be more advantageous because of its higher stability and lower restart frequency. Next, information models were investigated. A method of defining the Test Object and Associated Objects was developed to effectively implement concurrency control as well as permit the easy saving or retrieval of measurement information. Finally, application scenarios were proposed considering the appropriate order of resource settings and the standardized communication services. All of our research has confirmed that the OSI systems management standards are effective in offering interoperable interfaces for virtual path testing, a basic management operation in the ATM-based communication networks that are imminent. This paper provides the first step in achieving a B-ISDN testing environment managed by a standardized network management scheme.

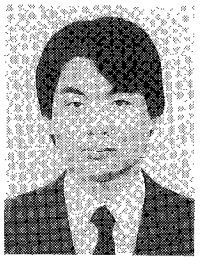
Acknowledgment

The authors gratefully acknowledge Kazumitsu Maki of NTT Transmission Systems Laboratories for his encouragement throughout this work.

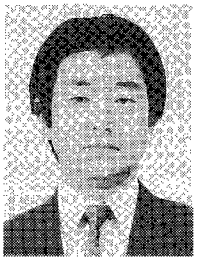
References

(1) Minzer, S. E., "Broadband ISDN and Asynchronous Transfer Mode (ATM)," *IEEE Commun. Mag.*, vol. 27, no. 9, pp. 17-24, Sep. 1989.
(2) Daddis Jr., G. E. and Torng, H. C., "A taxonomy of broadband integrated switching architectures," *IEEE Commun. Mag.*, vol. 27, no. 5, pp. 32-42, May 1989.
(3) Burgin, J. and Dorman, D., "Broadband ISDN resource management; The role of virtual paths," *IEEE Commun. Mag.*, vol. 29, no. 9, pp. 44-48, Sep. 1991.
(4) Tokizawa, I., Kanada, T. and Sato, K., "A new transport network architecture based on asynchronous transfer mode techniques," in *Proc. ISSLS '88*, (Boston, MA), Sep. 1988.
(5) Ohta, S., Sato, K. and Tokizawa, I., "A dynamically controllable ATM transport network based on the Virtual Path concept," in *Proc. GLOBECOM '88*, (Hollywood, FL), pp. 1272-1276, Nov. 1988.
(6) Sato, K., Ohta, S. and Tokizawa, I., "Broad-band ATM network architecture based on virtual paths," *IEEE Trans. Commun.*, vol. COM-38, no. 8, pp. 1212-1222, Aug. 1990.
(7) Addie, R. G., Burgin, J. L. and Sutherland, S. L., "Information transfer protocols for the broadband ISDN," in *Proc. GLOBECOM '88*, (Hollywood, FL), pp. 716-720, 1988.
(8) Kanayama, Y., Maeda, Y. and Ueda, H., "Virtual path management functions for broad-band ATM networks," in *Proc. GLOBECOM '91*, (Phoenix, AZ), pp. 1401-1405, Dec. 1991.
(9) Kanayama, Y., Matsunaga, H., Ohta, H., Maeda, Y. and Ueda, H., "Virtual path OAM functions for broad-band ATM networks," *IEICE Technical Report*, CS91-92, pp. 19-23, Oct. 1991.
(10) Bellcore, "Broadband ISDN operations: Framework generic criteria," *Framework Technical Advisory*, FA-NWT-001248, 1, Dec. 1991.
(11) Anderson, J. and Nguyen, M. D., "ATM-layer OAM implementation issues," *IEEE Commun. Mag.*, vol. 29, no. 9, pp. 79-81, Sep. 1991.
(12) Prycker, M. D., Desomer, M., Watteyne, M., Vandedrinck, J., Van Vyve, J. and Van Laethem, "An ATM switching

architecture with intrinsic multicast capabilities for the Belgian broadband experiment," in *Proc. ISS'90*, (Stockholm, Sweden), vol. V, pp. 111-118, May 1990.
(13) CCITT Draft Rec. M. 3010, "Principles for a telecommunications management network," 1991.
(14) ISO/IEC 10040, "Systems management overview," 1991.
(15) ISO/IEC DIS10164-12, "Test management function," 1991.
(16) Yoda, I., Sakae, K. and Fujii, N., "Configuration of local fiber optical network management system based on multiple manager systems environment," in *Proc. NOMS '92*, (Memphis, Tenn.), pp. 731-741, Apr. 1992.
(17) Rosenkrantz, D. J., Stearns, R. E. and Lewis II, P. M., "System level concurrency control for distributed database systems," *ACM Trans. Database Syst.*, vol. 3, no. 2, pp. 178-198, Jun. 1978.
(18) Franaszek, P. and Robinson, J. T., "Limitations of concurrency in transaction processing," *ACM Trans. Database Syst.*, vol. 10, no. 1, pp. 1-28, Mar. 1985.
(19) Bernstein, P. A. and Goodman, N., "Concurrency control in distributed database systems," Comput. Surv., vol. 13, no. 2, pp. 185-221, Jun. 1981.
(20) Agrawal, R., Carey, M. J. and McVoy, L. W., "The performance of alternative strategies for dealing with deadlocks in database management systems," *IEEE Trans. Software Eng.*, vol. SE-13, pp. 1348-1363, Dec. 1987.
(21) ISO/IEC 10164-1, "Object management function," 1991.
(22) CCITT SG VII contribution D478 "Study on enhancement to CMISE/CMIP on transaction processing capabilities," Apr. 1992.



Satoru Ohta received the B.E. and M.E. degrees in electronics engineering from the Tokyo Institute of Technology, Tokyo, Japan, in 1981 and 1983, respectively. Since 1983 he has been with NTT, where he worked on research and development of cross-connect systems, transmission network synthesis, and broadband ISDN. He is currently involved in research of transmission systems management at NTT Transmission Systems Laboratories. He is a member of the IEEE. He received the Excellent Paper Award in 1991 from IEICE.



Nobuo Fujii received the B.E. and M. E. degrees in applied physics from Osaka University in 1977 and 1979, respectively. In 1979, he joined NTT. Since then, he has been engaged in the research and development of the control system for digital cross-connect systems, the high-speed digital leased line system, and the telecommunications network operations system. He is currently a Senior Research Engineer at NTT Transmission Systems Laboratories. He is a member of the IEEE.