マルチメディア通信と分散処理56－4
（1992．7．9）

情報通信システム相互接続のための試験システムAICTSの開発

後藤憲一*　似内 聡*　高橋健一*　石幡吉則*　高橋 薫**
*㈱高度通信システム研究所（AIC）　**東北大学

開放型システム間相互接続技術の普及、発展とともに、その通信システムを構成する異機種端末間の相互接続試験の必要性が増している。本報告では、開放型システムの通信ソフトウェアに対する相互接続試験システムAICTS（Advanced Intelligent Communication System Laboratories' Inter Connectability Testing System）のアーキテクチャについて述べる。

さらに、その実現のために第一段階として開発を進めているAICTSプロトタイプのシステム構成、試験仕様の記述、試作状況などについて報告する。AICTSプロトタイプは、提案した相互接続試験システムAICTSの一部の構成モジュールを組み合わせたサブセットであり、トランスポート層クラス0を試験対象とし、適合性試験機能を実現するためのシステムである。

Design and Implementation of Interconnectability Testing System AICTS

Ken'ichi GOTOH*　Satoshi NITANAI*　Ken'ichi TAKAHASHI*　Yoshinori ISHIHATA*
Kaoru TAKAHASHI**

* Advanced Intelligent Communication System Laboratories（AIC）
** Tohoku University

* 6-6-3, Minamiyoshinari, Aoba-ku, Sendai 989-32, Japan
** 2-1-1, Katahira, Aoba-ku, Sendai 980, Japan

The interconnectability testing system between variety terminal equipments that have developed in conformity to the same OSI (Open Systems Interconnection) standard becomes very important following the diffusion of OSI. This paper describes the interconnectability testing architechture of AICTS, Advanced Intelligent Communication System Laboratories' Inter Connectability Testing System, and our approach to the development of prototype. The prototype of AICTS is composed of some components of interconnectability testing system AICTS, and is restricted in function to the conformance testing for transport class 0 protocol.

1．はじめに

開放型システムに対する国際標準(プロトコル仕様およびサービス定義)は、あるレベルで抽象化されたものであり、必ずしも実装に必要十分な細部規定は行っていない。そのため同一の国際標準に従って開発された情報通信端末であっても機種ごとに細部の仕様、設計は異なっている。このことは適合性試験[1],[2],[3]に合格していても相互接続ができない理由の一つと考えられる。そこで適合性試験を通過した端末での相互接続性の確認が必要となる。情報通信システムの大規模化・多様化にともなって異機種端末間の相互接続試験に対する認識が高まってきており、研究例もいくつか報告されている[4],[5],[6],[7]が、統一的な相互接続試験アーキテクチャは未だ確立されていない。

そこで我々は、開放型システムを構成する通信ソフトウェアを試験対象とし、試験の自動化、汎用化を目指した相互接続試験システム AICTS(Advanced Intelligent Communication System Laboratories' Inter Connectability Testing System)の構想をまとめ、その開発を進めている[8]。本報告では、試験アーキテクチャ、システム構成、プロトタイプの試作状況について述べる。

2．相互接続試験システムＡＩＣＴＳアーキテクチャ

2．1 相互接続試験の定義

情報通信システムの相互接続性を高める試験としては、相互接続試験と適合性試験が考えられる。相互接続試験とは、実利用環境あるいはそれと同等の環境において相互に接続された(N)層試験対象実装同士が矛盾なく動作し、ユーザの要求するサービスを提供できるかどうかを確認するための試験とする。

これに対して適合性試験は、特定のシステム、すなわちプロトコルの標準実装としてふるまう試験システムと試験対象実装との接続性を観測することにより標準に対する適合性を判定するものである。適合性試験においては、下位ネットワークの動作は常に保証されていることを前提にプロトコルデータ単位 (PDU:Protocol Data Unit)の文法や送達手順を重視した試験項目が実行される。しかし、実利用環境においてはネットワークの状態は、常に変動しており障害も生じる。変動するネットワークにおける実装の相互接続性を高めるには、適合性試験とは別の観点で試験を行う必要がある。

なお、運用開始時に行われるシステム全体の相互運用性試験と相互接続試験とは後者が試験対象を (N)- エンティティの動作に絞っている点で異なる。

2．2 相互接続試験法の概念モデル

相互接続試験は、試験システムが介在しながら試験対象実装(IUT :Inplementatiom Under Test)間の接続性について試験・評価を行うものである。試験システムに対しては、 IUT の制御・観測を行えるとともに IUT 間を流れるデータを可能な限り透過的に扱いながら試験を行えることも要求される。これを実現するための理想的な相互接続試験システムの概念モデルを図1に示す。

ここで、IUTは、OSI7層モデルの(N)層の単一層エンティティ、下位層は誤りの発生しない完全なサービス提供媒体であると仮定する。

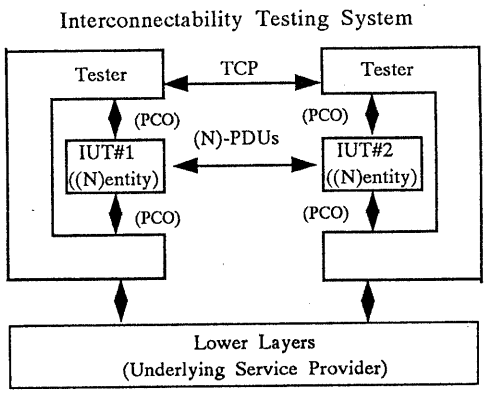


図1 相互接続試験の概念モデル

相互接続試験では、(N)層サービスを制御・観測することが重要であるため、サービスアクセス点(SAP:Service Access Point)を制御・観測点(PCO:Point of Control and Observation)に割り当てるのが最も適切と考えられる。また、抽象サービスプリミティブ(ASP: Abstract Service Primitive)およびPDUの送受信タイミングを観測するためIUTの上位と下位、両方にPCOを置く必要があり、従って、相互接続試験のテスタ(Tester)はIUTを包み込むような形態となる。また、テスタ間では、試験調和手続き (TCP: Test Coordination Procedures)を用いて試験対象システム(SUT: System Under Test)および試験システム相互の動作の協調を行う。

2．3 ＡＩＣＴＳ開発の基本方針

相互接続試験システムAICTS開発の基本方針を次に示す。

(1) 試験システムの介在による影響を最小限とし相互接続試験対象相互の関係を可能な限り実運用状態に近づける。

(2) 試験項目の設定、試験実行、判定などの試験機能の自動化をはかる。

(3) 試験機能をモジュール化し、異なる試験対象プロトコルに対するプログラム部品の再利用可能化をはかる。

(4) 試験の仕様記述である試験スイートは、国際標準試験仕様記述言語TTCN(Tree and Tabular Combined Notation)を用いて相互接続試験用のものを開発する。

2．4 システム構成

相互接続試験法の概念モデルを試験システムとして実現するためのAICTS論理構成を図2に示す。

AICTS は、テストマネージャ、アクティブテスタ、パッシブテスタで構成し、機能別に分散配置する。試験環境は、テストマシ

ン(テストマネージャと2つのアクティブテスタ)、2つの試験対象システムSUT(パッシブテスタとIUT)#1、#2および下位層サービスから構成する。なおAICTSを構成する各部(テストマネージャ、アクティブテスタ、パッシブテスタ)の機能を表1に示す。

表1 ＡＩＣＴＳ各部の機能

モジュール	主な機能
テストマネージャ	1．アクティブテスタに対する試験実行指示 2．ＩＵＴ間のＰＤＵ中継・モニタリング 3．ＩＵＴ間の相互接続性の判定 4．相互接続失敗時の要因解析の支援
アクティブテスタ	1．テストマネージャの指示による試験ケース選択と実行 2．パッシブテスタとのＴＭＭ送受信 3．ＩＵＴ間のＰＤＵ中継 4．例外発生模擬用の異常ＰＤＵ発生
パッシブテスタ	1．アクティブテスタとのＴＭＭ送受信 2．ＩＵＴ上位ＳＡＰへのアクセス、サービスプリミティブ交換

相互接続試験システムAICTSでは、試験実行自動化の観点から試験項目の設定、試験実行、判定などの試験の中核となる機能をテストマネージャと2つのアクティブテスタとしてテストマシンに集約する。またSUTの構成を簡易なものとするためパッシブテスタには、最小限の機能を搭載することとして、試験対象実装IUTの上位SAPの制御・観測機能とアクティブテスタとの動作調和のための試験管理メッセージ通信機能を配し、IUTとともに試験対象システムSUTを構成する。

テストマシン、SUT間の試験調和手続きは、試験管理プロトコル(TMP:Test Management Protocol)に従ってテストマネージャ、アクティブテスタ、パッシブテスタ間で試験管理メッセージ(TMM:Test Management Message)を通信し合うことにより行う。従来の試験管理メッセージ通信は、試験データが流れる試験チャネルを用いているが、本システムでは、別に試験管理チャネルを設けて試験チャネルと試験管理チャネルを分ける。この方法により使用する論理回線は増えるが試験管理の信頼性向上、試験シーケンスの自由度向上の大きな効果を期待できる。正常系試験では、実運用状態と同じくIUT間のコネクションはテストマシンを経由せずに直接確立する。不正な(N)-サービスプリミティブの発行に対する反応をみる異常系試験では、PDUのモニタリングを行うため、アクティブテスタ、テストマネージャを経由させる。

2．4 試験管理プロトコル

試験管理プロトコルTMPに基づいて通信されるTMMの種類と機能を図3に示す。

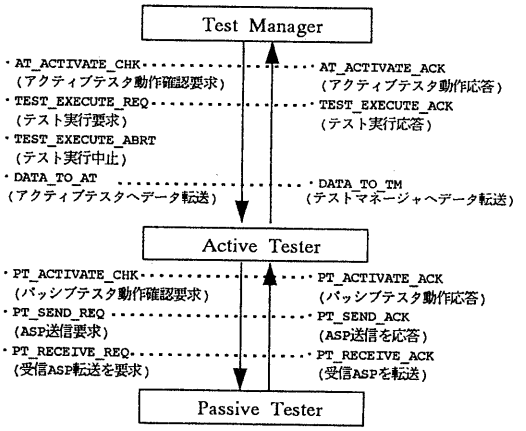


図3 試験管理メッセージＴＭＭの種類と機能

試験調和手続きは、以下の手順概要による。

(1) テストマネージャが試験ケースに記述された手順に従って、試験項目識別番号をパラメータとして含む試験指示イベントをアクティブテスタに送信する。

(2) アクティブテスタは該当する試験項目に従ってパッシブテスタに試験手順を指示する。

(3) パッシブテスタは、IUTの上位SAPの制御・観測を行う。

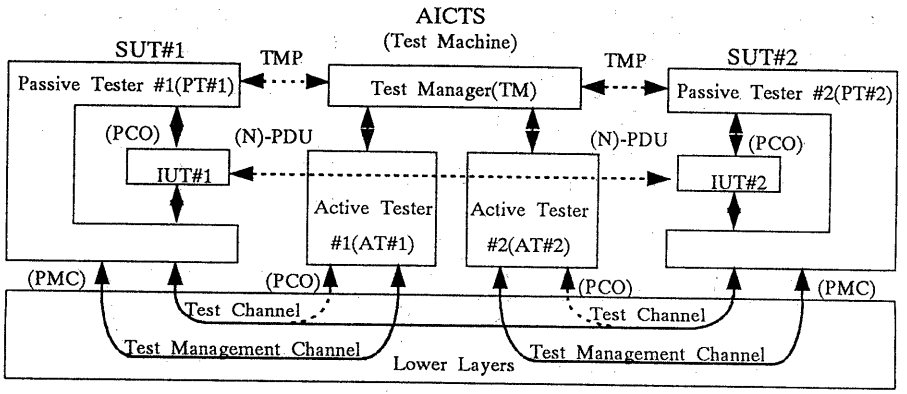


図2 相互接続試験システムＡＩＣＴＳの論理構成

(4) アクティブテスタは該当する試験項目に従ってIUTの下位SAPの制御・観測を行う。

(5) パッシブテスタは、アクティブテスタの指示によりIUTの上位SAPの制御・観測結果をTMMとしてアクティブテスタに送信する。

(6) アクティブテスタは、収集したIUTの上位および下位SAPの制御・観測結果をTMMとしてテストマネージャに送信する。

(7) テストマネージャは、一方のアクティブテスタから受けたTMMをもう一方のアクティブテスタにTMMとして中継、送信し、同時にデータのモニタを行う。

(8) 試験結果の判定は、IUTの上位、下位SAPの制御・観測結果とデータのモニタ結果およびプロトコル仕様、サービス仕様からテストマネージャが行う。

3. AICTSプロトタイプの設計

相互接続試験システムAICTSは、前述のように2組のアクティブテスタとパッシブテスタおよびテストマネージャから構成される。従って1組のアクティブテスタとパッシブテスタは全体のサブセットとして扱うことができる。そこで、このサブセットをAICTSのプロトタイプとして位置付け下記の開発方針を立てた。

3. 1 開発方針

(1) 試験機能: 適合性試験

(2) 試験対象プロトコル: トランスポート層クラス0

(3) 下位ネットワーク: X.25パケット通信網相当

(4) 試験スイート: OSTC適合性試験スイート[9] 相当のものを開発

(5) 試験実行方式: 試験スイートインタプリタ方式

プロトタイプの試験機能は、システム構成を考慮してアクティブテスタに試験ケースの選択と実行機能などを持たせて、適合性試験が行えるものとする。

試験対象プロトコルは、標準化が完了しておりプロトコルとしてシンプルなトランスポート層とする。

3. 2 システム構成と機能

AICTSプロトタイプの構成を図4に示す。本システムはアクティブテスタとパッシブテスタに機能分割され、各テスタは、さらに複数のモジュールから構成される。アクティブテスタおよびパッシブテスタは次の機能を持つ。

(1) アクティブテスタの機能
- 試験ケースの選択と実行
- TMMによるパッシブテスタの制御
- 試験結果の判定と解析支援
- 報告書の作成

(2) パッシブテスタの機能
- アクティブテスタからの試験制御用のTMM受信
- IUT上位SAPとのサービスプリミティブ交換
- サービスプリミティブ交換結果をTMMによりアクティブテスタに送信

SUTの構成は、プラットフォームとなるマシン環境に依存する。そこで、パッシブテスタでは、IUTとのインタフェース機能をインタフェースモジュールとして独立させ、異なるマシン環境の場合には、インタフェースモジュール変更のみですむ様にする。

なお相互接続試験システムに拡張する場合は、アクティブテスタの機能のうち報告書の作成、試験ケースの選択と実行、解析支援はテストマネージャに持たせる予定である。

3. 3 試験実行フロー

試験の実行フローを図5に示す。システムの操作は、マウスまたはキーボードを使って行う。

(1) 環境設定では、アクティブテスタの動作に必要な入出力ファイルや装置環境等の情報を入力する。

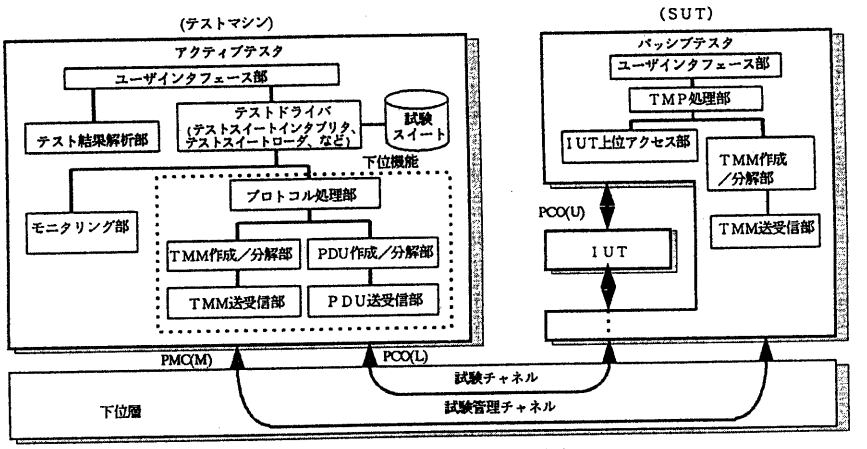


図4 AICTSプロトタイプの構成

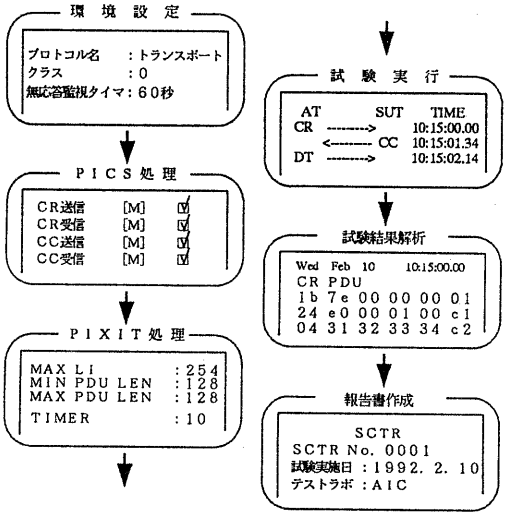


図5 試験の実行フロー

(2) PICS処理では、プロトコル実装適合性宣言(PICS : Protocol Implementation Conformance Statement)に記載されたデータの入力を行う。

(3) PIXIT処理では、試験用プロトコル実装補助情報(PIXIT : Protocol Implementation Extra Information for testing)に記載されたデータの入力を行う。PICS、PIXIT入力データをもとにIUT が実装している機能項目に漏れやプロトコル実装違反がないかチェックを行う。

(4) 試験実行においては、試験ケースの選択、スケジュール作成を行い、試験対象システムの試験開始準備後にスケジュールに基づいて試験を実行する。試験の判定結果は、通過、失敗、不定のいずれかとして試験ケース毎に自動的に判定され、試験実行ログファイルに記録される。また試験データは、トレースデータファイルに記録、保存される。

なお、試験実行におけるデータの送受信は、テストスイートローダによってテストマシンのメモリに展開された実行型試験スイートをテストスイートインタプリタが参照して、各機能部を制御しながら行う試験スイートインタプリタ方式を用いている。

(5) 試験結果解析では、ファイルに保存された試験実行ログおよび試験実行中のデータの流れをリアルタイムで記録したトレースデータを画面表示して参照し、失敗、不定の原因解析、再試験の要否などの試験結果の解析を行う。

(6) 試験結果は、プロトコル適合性試験報告(PCTR : Protocol Conformance Test Report) およびシステム適合性試験報告(SCTR: System Conformance Test Report)としてまとめられ、出力される。

3．4 抽象試験スイート

我々は、AICTS プロトタイプの抽象試験スイート (ATS:Abstract Test Suite)の記述法としてTTCN.GR(TTCN Graphical Form)を採用する。TTCN は、適合性試験の方法と枠組みの規定に従って抽象試験スイートを記述するための言語として国際標準化されたものである [1],[2]。トランスポート層クラス0用ATSは、種々のものが知られているが、欧州の事実上の標準であるOSTC(Open Systems Testing Consortium)[9] により開発されたものと同等のものを開発することとした。

AICTS プロトタイプ用 ATS では、試験管理チャネル (TMC:Test Managenent Channel) の制御・観測が必要となることからアクティブテスタとパッシブテスタ間に試験管理制御点 (PMC: Point of Management and Control)を設ける。抽象試験仕様の記述は、IUTの上位および下位の制御観測点PCO(U およびL) とPMC(M) に対して行う。

図6に試験管理チャネルのコネクションの確立とパッシブテスタの動作確認のための試験項目記述例を示す。記述方法は、次のとおり。

M!ASP名(TMM名): PMCにおいてTMM名をデータとして含むASP名を送信する。

M?ASP名(TMM名): PMCにおいてTMM名をデータとして含むASP名を受信する。

ここでTMCはMで表わしている。また'!'はアクティブテスタからの送信を表わし、'?'はアクティブテスタでの受信を表わす。

Test Ref : Test ID : TMM1 / Preamble	Purpose : Test Management channelのコネクションの確立とPTの動作確認を行う。			
Defaults Ref : Def 1				
Behaviour Description	Label	Refer	Comments	Result
M ! N_CON_REQ Start (Y) M ? N_CON_CONF Cancel (Y) M ! N_DATA_REQ(PT_ACT_CHK) Start (Z) M ? N_DATA_IND(PT_ACT_ACK) Cancel (Z) ? Timeout (Z) +Pos1 / Postamble ? Timeout (Y) +Pos1 / Postamble			(1) (2) (3) (4) (5)	

図6 テスト管理チャネルTMC確立の試験項目記述例

図6の内容の概略を以下に示す。

(1) TMC 用のネットワークコネクション接続要求をアクティブテスタから送信する。

(2) TMC 用のネットワークコネクション接続確認をアクティブテスタで受信する。

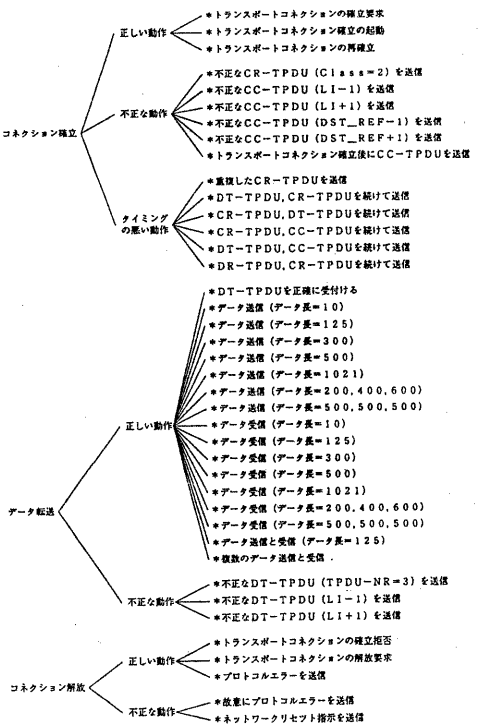
(3) TMC を用いてパッシブテスタの動作確認データをアクティブテスタから送信する。

(4) TMCを用いてパッシブテスタの動作状態の通知データをアクティブテスタで受信する。

(5) TMC用のネットワークコネクションの解放を行う。

表2にトランスポート層クラス0の試験スイートの試験目的の分類を示す。試験ケース数は40項目である。また、図7には、トランスポート層クラス0のトランスポートコネクションの確立試験の抽象試験スイート例を示す。

表2 トランスポート層クラス0試験目的分類



Test Ref : TR0/0A1/B01/001 Test ID : Defaults Ref : Def 2	Purpose : トランスポートコネクションの確立要求を受け付けることを確認する。			
Behaviour Description	Label	Refer	Comments	Result
+TMM1 / Preamble			(1)	
+TXU1 / Preamble			(2)	
L ! N_DATA_REQ(CR)		CR1	(3)	
Start (A)				
L ? N_DATA_IND(CC)		CC21	(4)	pass
Cancel (A)				
+Pos2 / Postamble			(5)	
? Timeout (A)				fail
+Pos2 / Postamble			(5)	

Comments :
(1) Test Management channelのネットワークコネクションの確立を行う
(2) Test channelのネットワークコネクションの確立を行う
(3) ATからCR－TPDUをIUTへ送信
(4) IUTからのCC－TPDUをATで受信
(5) Test channelとTest Management channelのネットワークコネクションの解放を行う

図7 抽象試験スイート例

3．5 実行型試験スイート

抽象試験スイートATSは、そのままの形では、試験システムによる試験の実行には適さない。そこで、このATSの抽象的な部分を具現化し、プログラムで操作を可能にした実行型試験スイート(ETS:Executable Test Suite)を作成する。このETSは、TTCN.GRで記述されたATSを参照し、エディタを用いてテキストファイル形式で作成し保存する。試験では、テストスイートローダがそのファイルを試験実行マシンのメモリ上に展開し、それをテストスイートインタプリタが操作しながら各部の制御を行い試験を実行する。

(1) ETSの仕様

ETSは、通信手順を記述する際に一般に用いられる状態遷移表の構造を持つ。ETSの記述例を図8に示す。

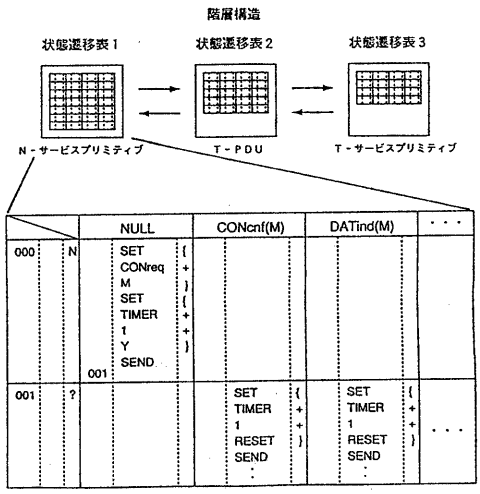


図8 ETSの記述例

状態遷移表の横軸は外部からのイベント、縦軸は状態番号を表わし、表の各欄には、その状態で実行するアクションを表記する。なおETSは、複数のレイヤの動作に関係するので、複数の状態遷移表により構成する。すなわちイベントの発生する場所の違いによりそれぞれに状態遷移表を持ち、階層的な関係を有している。この方法によりレイヤ別に試験手順、すなわち試験時の通信手順が記述できるので、レイヤ毎の着目したい点の動作がより明確に記述できる。これら状態遷移表は、下位レイヤから順に状態遷移表1(ルートの状態遷移表)、状態遷移表2、状態遷移表3、・・・・・、状態遷移表nと番号づける。

例えば、トランスポート(T)層では、トランスポートエンティティ、ネットワーク(N)サービス、トランスポートサービスと複数のレイヤの動作に関係するので、ETSは、次の3つの状態遷移表からなる。

1) N-サービスプリミティブの入力用

2) T-PDUの入力用

3) T-サービスプリミティブの入力用

また、テストスイートインタプリタに可搬性を持たせるため、状態遷移表内では、下位機能が実行するアクションは全て記号として表わしてある。テストスイートインタプリタは、この記号が何を意味するかを知らず、状態遷移表の指示に従いその記号を下位機能に渡すだけである。

例えば図8の状態遷移表1(N-サービスプリミティブ)の場合、テストスイートインタプリタは、初期状態「000」、入力イベント「NULL」に於いて次の動作を行う。

1) 試験管理チャネル(記号'M')について、コネクション確立要求の(N)-サービスプリミティブ(記号"CONreq")を発行することを下位機能に指示する。

2) 下位機能に対して、1番目のタイマ(記号"TIMER"、'1')を'Y'秒間動作させることを指示する。

3) 次の状態「001」へ遷移し、記号「?」で示されるイベント入力待ち状態となる。

(2) ETSに基づく試験実行手順

試験実行手順を図9に示す。

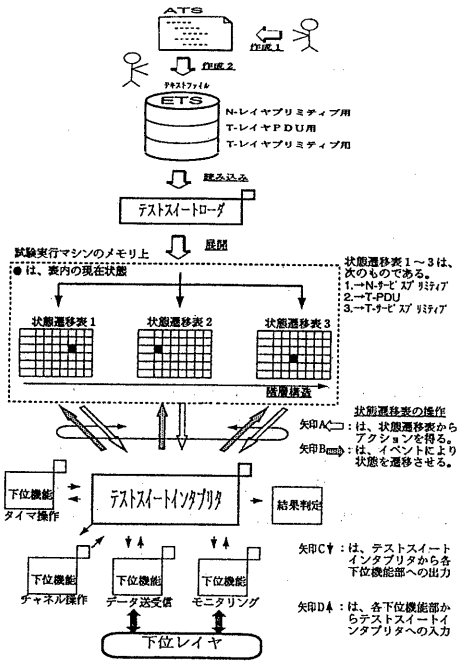


図9 試験実行手順

試験の実行は、下記のとおり行われる。

1)ATSをTTCN.GRにより記述、作成する。(作成1)

2)ATSを参照し、エディタでテキストファイル形式のETSを作成する。(作成2)

3)テストスイートローダは、指定されたテキストファイル形式のETSを読み、試験実行マシンのメモリ上に状態遷移表を階層的に分割して展開する。(読み込み)、(展開)

4)テストスイートインタプリタは、メモリ上の状態遷移表を参照し、現在行うべきアクションを意味する記号を得る。(矢印A)

5)テストスイートインタプリタは、下位機能部(タイマ操作、試験チャネルおよび試験管理チャネルの操作、データ送受信など)へ渡す。(矢印C)

6)下位機能部は、指示されたアクションを起こし、その結果をテストスイートインタプリタへ通知する。(矢印D)

7)テストスイートインタプリタは、下位機能部からのイベントにより、階層的に分割されているそれぞれの状態遷移表の状態番号を次の状態番号に遷移させる。(矢印B)

8) 4)から7)までの処理を一つの試験ケースが終了するまで繰り返す。

ここで、7) 項についてどの状態遷移表の状態遷移表番号を遷移させるかは、階層のルートにある状態遷移表1から順に見て行き判断する。例えば、状態遷移表1内に「状態遷移表2を参照せよ」と記述があれば、状態遷移表1と2の状態番号を遷移する。遷移表2から3へも同様である。

4. AICTSプロトタイプの試作

4. 1 試験実行環境

AICTSプロトタイプの試験実行環境を図10に示す。

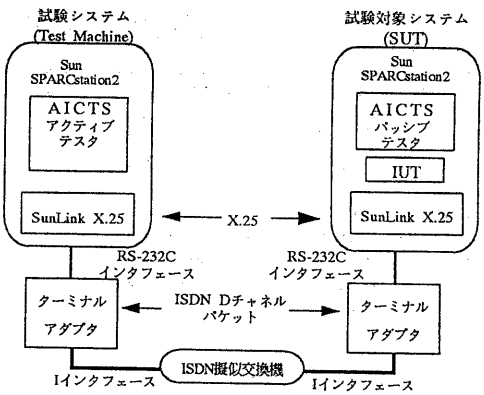


図10 試験実行環境

AICTSプロトタイプのソフトウェアは、C言語を用いてプログラミングした。テストマシンおよびSUTのプラットフォームは、ワークステーションSun SPARCstation2を使用し、下位層の模擬通信網は、DSUとISDN擬似交換機を組み合わせて構成した。模擬通信網内はISDNパケット通信モードで動作する。ワークステーション

間は、通信ソフトウェアSun Link X.25 Ver.6.0を使いX.25プロトコルにより通信を行う。

IUTについては、種々のIUTを模擬できるIUTシミュレータを開発した。IUTシミュレータは、トランスポート層クラス0のエンティティを実現するソフトウェアであり、各種パラメータの値を設定可変とし、AICTS プロトタイプの動作試験とデバッキングも行えるようにした。

4．2 開発状況

AICTS プロトタイプは、試験実行に必要な主要な部分については試作済みであり、現在、残りの試験結果解析部などの開発を進めている。プログラムの規模は、現在、約20 キロステップであるが最終的には、30キロステップ程度となる見込みである。

システム操作画面例を図11 に示す。これは、試験を実行した場合に表示される画面であり複数のウィンドウが表示される。ウィンドウ「TEST CASE INTERPRETER」には、左枠にマウスの操作状況、中央枠に試験実行中のシステムメッセージ、右枠に試験スイートの実行結果が表示される。ウィンドウ「 TESTING MONITOR」の左側には、アクション実行(もしくはイベント入力)時間、送受信T-PDU(もしくはTMM)名、送受信N-サービスプリミティブ名が表示される。また、中央部に表示される「 -----> 」、「====>>」などはアクティブテスタとSUT間のデータの送受信方向を示す。右側には、SUTが送受信したN-サービスプリミティブ名、T-PDU(もしくはTMM)名、IUTとパッシブテスタ間の送受信T-サービスプリミティブ名が表示される。

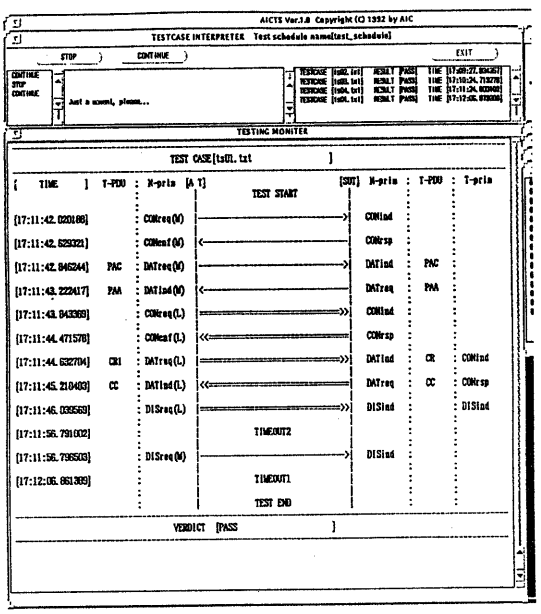


図11 システム操作画面例

5．まとめと今後の予定

相互接続試験システム AICTS のこれまでの開発結果から AICTS アーキテクチャの基本機能を確認することができた。現在、結果報告書作成機能などの残りの部分の開発と開発済の試験実行機能の評価を進めている。

今後は、AICTS プロトタイプを機能拡張し、相互接続試験機能を実現するためのテストマネージャの開発、試験スイート生成支援ツールの開発などを進める予定である。

なお、試験スイートの検証問題および今後予定している上位層プロトコル用試験への機能拡張の際に必要となるマルチパーティ試験の構成法については、今後の課題と考えている。

謝 辞

本研究に対して有益なご助言を頂いた東北大学野口正一教授、白鳥則郎教授、AIC緒方秀夫常務に深謝いたします。

参考文献

[1] CCITT X.290: "OSI Conformance Testing Methodology and Framework for Protocol Recommendations for CCITT Applications", (1988)

[2] ISO/IEC 9646: "Information technology - Open Systems Interconnection - Conformance testing methodology and framework",(1991)

[3] 若杉: "OSI適合性検証試験の現状",情報処理学会マルチメディア通信と分散処理研究会、52-21(1991.9)

[4] G.Bochmann,R.Dssouli,J.R.Zhao: "Tace Analysis for Conformance and Arbitration Testing", IEEE Tansactions on Software Engineering,Vol.15,N0.11,pp1347-1356(1989.11)

[5] O.Rafig and R.Chastanet: "From conformance testing to interoperability testing", Proceedings of IFIP TC6 Third International Workshop on Protocol Test Systems(1990)

[6] L.Lenzini,EZoccolini: "Interoperability tests on OSI products in the framework of the OSIRIDE-Intertest initiative", Computer Networks and ISDN Systems 24(1992)

[7] 中井、高橋、白鳥、野口: "相互接続試験アーキテクチャの構成法",電子情報通信学会技術研究報告、SSE90-58(1990.7)

[8] 石幡、後藤、似内、高橋、高橋: "相互接続試験システムICTSの基本構想",情報処理学会第43回全国大5T-08,(1991.10)

[9] OSTC: "OSTC ABSTRACT TEST SUITE TRANPORT CLASS 0 ",(1991.1)