CSCI 234

Design of Internet Protocols: Protocol Testing

George Blankenship

Outline

- Testing of Telecommunications Systems
- Conformance Testing
- Testing Notation (TTCN)
- Test Equipment
- Interoperability Testing
- Automated Validation
- Traffic Theory

Testing of the Telecommunications Systems

- Functional Tests
- Formal System Tests against system level requirements
- Protocol Verification
- Performance/Load Tests
- Interoperability Tests- IOT (Inter-working Tests)
- Automatic Test Suite

Protocol Testing

- **Conformance Testing**
  - Performed utilizing test equipment
  - Based on Conformance Test Suites/Cases specified by standard body
    - Very detailed in terms of protocol implementation verification

  (-) Not good testing performance/load issues
- **Inter-working Trials**
  - Testing of real equipment inter-working

  (+) Good for performance/load testing

  (-) Hard to completely verify protocol

Protocol Testing George Blankenship 4

Conformance Testing

- Based on ISO/IEC 9646 (or X.290)
  - *"Framework and Methodology of Conformance Testing of Implementations of OSI and CCITT Protocols."*
- Abstract Test Suites (ATS) consisting of Abstract Test Cases
- Test Cases defined using "Black Box" model
  - Observing external interfaces
- Provide basis for
  - Generic Test Tools
  - Methods for verification of Telecom Standards and Protocols

Protocol Testing George Blankenship 5

ITU Conformance Testing Standards

- x.290 OSI conformance testing methodology and framework for protocol recommendations
- x.291 Abstract test suite specification
- x.292 The Tree and Tabular Combined Notation (TTCN)
- x.293 Test realization
- x.294 Requirements on test laboratories and clients for the conformance assessment process
- x.295 Protocol profile test specification
- x.296 Implementation conformance statements

Protocol Testing George Blankenship 6

Conformance Testing

- Verification ensures that protocols are conforming to Standard Requirements
- PICS - Protocol Implementation Conformance Statement
  - Information: provided by protocol implementator
  - Implementation info: optional items, restrictions…
  - Method: PICS questionnaire form provided by standard body
- PIXIT - Protocol implementation extra information
  - Physical configuration of unit under test:
  - e.g.. Telephone numbers, socket numbers ...

Protocol Testing George Blankenship 7

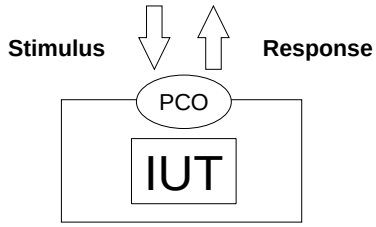
Tree and Tabular Combined Notation TTCN

- Part of ISO/IEC 9646 (X.290)
- Language for formal definition of test cases
- Each test case is **an event tree** in which external behavior such as:

  "**If** we send the message ' *connect request*' **then** *'connect confirm*' or ' *disconnect indication*' will be received"

Protocol Testing George Blankenship 8

TTCN Tests Reactive Systems

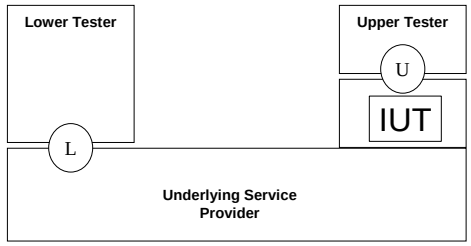


PCO = Point of Control and Observation

IUT = Implementation Under Test

Protocol Testing George Blankenship 9

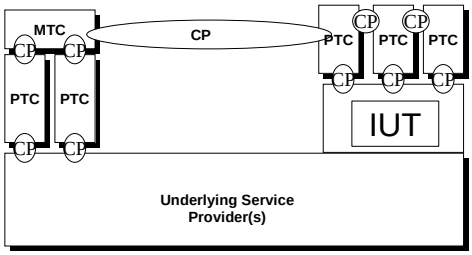
A Typical Test Architecture



IUT = Implementation Under Test

Protocol Testing George Blankenship 10

Parallel Test Architecture



MTC = Main Test Component
PTC = Parallel Test Component
CP = Coordination Point

Protocol Testing George Blankenship 11

TTCN Formats

- TTCN is provided in two forms:
  - **graphical** form (TTCN.GR) suitable for human readability
  - **machine** processable form (TTCN.MP) suitable for transmission of TTCN descriptions between machines and possibly suitable for other automated processing.

TTCN: Tree and Tabular Combined Notation

Protocol Testing George Blankenship 12

Main Components of TTCN

- Test Suite Structure
- Data declarations
  - TTCN data types
  - ASN.1 data types
- Constraints
  - Dynamic behaviour trees (Test Cases)
  - Send, Receive, Timers, Expressions, Qualifiers
  - Test Steps

Test Suite Structure

- Four major parts
  1)suite overview part
  2)declarations part
  3)constraints part
  4)dynamic part

Suite Overview Part

- A documentary feature: indexes and page references
  - heritage as a paper-oriented language
- Contains:
  - table of contents
  - description of the test suite
  - Purpose
    - document the test suite
    - Increase: clarity, readability
- Quick overview of the entire test suite (optional)

Test Suite Overview Part

Test Suite Structure			
Suite Name : *SuiteIdentifier*			
Standards Ref : *Free Text*			
PICS Ref : *Free Text*			
PIXIT Ref : *Free Text*			
Test Method(s) : *FreeText*			
Comments : *[FreeText]*			
Test Group Reference	**Selection Ref**	**Test Group Objective**	**Page No.**
TestGroupReference	*[SelectExprIdentifier]*	*FreeText*	*Number*
Detailed Comments: *[FreeText]*			

Protocol Testing George Blankenship 16

Suite Declarations Part

- To declare
  - types
  - variables
  - timers
  - points of control and observation (PCOs),
  - test components
- Graphical table is used for declarations

Protocol Testing George Blankenship 17

TTCN Data Types

- No equivalent to pointer types.
  - **THEREFORE**: types cannot be directly or indirectly recursive.
- Two structured types are specific:
  - protocol data unit (PDU) – data packets sent between peer entities in the protocol stack
  - abstract service primitive (ASP) -- to embed a PDU for sending and receiving

Protocol Testing George Blankenship 18

Example of ASN.1 Type Definition

ASN.1 Type Definition	
Type Name	: DATE_type
Comments	: To illustrate the structure of ASN.1 type definitions

Type Definition

```
SEQUENCE {
        day     DAY_type,
        month MONTH_type,
        year    YEAR_type
    }
-- local DAY_type
DAY_type ::= INTEGER {first(), last(31)}
-- MONTH_type and YEAR_type are defined ASN.1 Type Definitions tables
```

Protocol Testing George Blankenship 19

Example of TTCN Type Definition

TTCN Type Definition		
Type Name	: DATE_type	
Comments	: To illustrate the structure of TTCN type definitions	
Parameter Name	Parameter Type	Comments
day	DAY_type	
month	MONTH_type	
year	YEAR_type	

Detailed Comments:

Protocol Testing George Blankenship 20

Variable Declarations

Test Suite Variable Declarations			
Variable Name	**Type**	**Value**	**Comments**
state	IA5String	"idle"	Used to indicate the final stable state of the previous Test Case, if any, in order to help determine which preamble to use.

Protocol Testing George Blankenship 21

Constraints Part

- Constraints used to describe values sent or received
- Instances used for sending must be complete
- Instances used for receiving may be left incomplete (using wild cards, ranges and lists).

PDU Constraint Declaration

Constraint Name	: C1	
PDU Type	: PDU_A	
Derivation Path	:	
Comments	:	
Field Name	**Field Value**	**Comments**
FIELD1	(4 .. INFINITY)	
FIELD2	TRUE	
FIELD3	"A STRING"	

Protocol Testing George Blankenship 22

Possible Constraint

ASN.1 PDU Type Definition

PDU Name	: XY_PDU
PCO Type	:
Comments	:

Type Definition

```
SET     {  field_1 INTEGER OPTIONAL,
           field_2 BOOLEAN,
           field_3 INTEGER OPTIONAL,
           field_4 INTEGER OPTIONAL  }
```

ASN.1 PDU Constraint Declaration

Constraint Name	: CONS1
PDU Type	: XY_PDU
Derivation Path	:
Comments	:

Constraint Value

```
{  field_1   5,
   field_2   TRUE,
   field_3   3
}
```

- *field_4 is not specified=>omitted when sending*
  If identifier field_3 was not used it would be ambiguous whether 3 was the value of field_3 or field_4, since both are OPTIONAL.

Protocol Testing George Blankenship 23

Dynamic Part

- Definition of tests
- Contains Test Suite
  - ***test groups***
    - all test cases concerning connection establishment and transport
  - ***test steps***
    - test events, similar to a subroutine or procedure in other programming languages
  - ***test events***
    - the smallest, indivisible unit of a test suite.
    - sending or receiving a message operations for manipulating timers

Protocol Testing George Blankenship 24

TTCN Behavior Tree

- Suppose that the following sequence of events can occur during a test whose purpose is to establish a connection, exchange some data, and close the connection.

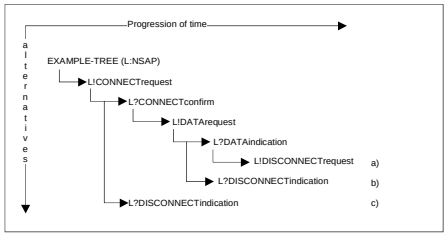
  a) CONNECTrequest, CONNECTconfirm, DATArequest, DATAindication,DISCONNECTrequest.

- Possible alternatives to "valid behavior" are

  b) CONNECTrequest, CONNECTconfirm, DATArequest,DISCONNECTindication.

  c) CONNECTrequest, DISCONNECTindication.

TTCN Behavior Tree



Dynamic Behaviour Table

A TTCN behaviour tree:

Test Step Dynamic Behaviour					
Test Step Name : TREE_EX_1(L:NSAP)					
Group : TTCN_EXAMPLES/TREE_EXAMPLE_1/					
Objective : To illustrate the use of trees.					
Default : NOTE – This example can be simplified by using Defaults.					
No.	**Label**	**Behaviour Description**	**Constraints Ref**	**Verdict**	**Comments**
1		L!CONNECTrequest	CR1		Request
2		L?CONNECTconfirm	CC1		Confirm
3		L!DATArequest	DTR1		Send Data
4		L?DATAindication	DTI1		Receive Data
5		L!DISCONNECTrequest	DSCR1	PASS	Accept
6		L?DISCONNECTindication	DSCI1	INCONC	Premature
7		L?DISCONNECTindication	DSCR1	INCONC	Premature

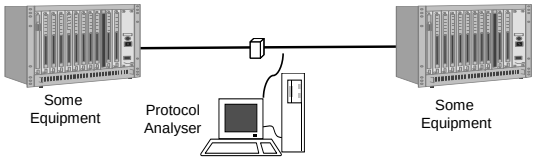
TTCN MP Form

```
$Begin_TestCase
$TestCaseId TC43__S0_17
$TestGroupRef V52NWKAN/BCC/BV/
$TestPurpose /* On receipt of an AUDIT message containing an UP_ID IE,
the IUT shall send an AUDIT COMPLETE message containing the
connection_incomplete IE "incomplete normal". */
$DefaultsRef DEF_BCC_ANY_BODY
$BehaviourDescription
$BehaviourLine
$LabelId
$Line [0] +Preamble_bcc_AN20
$Cref
$VerdictId
$Comment /*  */
$End_BehaviourLine
$BehaviourLine
$LabelId
$Line [1] +STEP_bcc_init
$Cref
$VerdictId
$Comment /*  */
$End_BehaviourLine
$BehaviourLine
$LabelId
```


Various Test Equipment

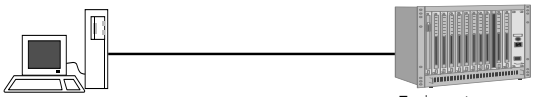
- Numerous Test Equipment supports the TTCN
- Protocol Analyzers
- Protocol Simulators
- Protocol Emulators

Protocol Analysers (Monitors)



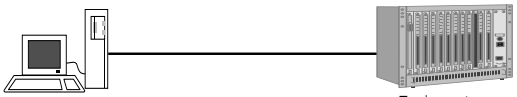
- Monitors of the 'live' communication between equipment and interprets messages in human readable form.
- Same H/W may support monitoring of different protocol types
- It may provide some statistical analysis capability
- It may have advanced search/filtering capabilities.
- You may define event on which logging would start or stop.

Protocol Simulators



- Allows Users to write a scripts (similar to TTCN) which will send a messages and react on received messages
- It can be used for testing various protocols
- It may be quite big effort to write required amount of scripts
- Suppliers often provide a set of test scripts targeting particular protocol

Protocol Emulators



- Emulates operation of equipment (not full implementation)
- No script writing required
- It provides some control of behavior and configuration
- Not flexible as simulator
- Usually one box can contain Protocol Analyser, Simulator and Emulator

Interoperability Testing (IOT)

- Testing what users may expect from product, which may include more then demanded by standards: ***performance***, **robustness** and **reliability**
- Proper functioning where product is finally installed
- Interoperability with other applications
- IOT is usually performed in addition to Formal Protocol Conformance Testing

Pros & Cons of IOT

- **Potential IOT Problems**
  - May not cover all possible protocol scenarios
  - Difficult to reproduce detected problems
  - It is difficult to control test environment for problem reproduction
- **Pros of IOT**
  - Standard may be ambiguous (unique implementations)
  - Performance/Load aspects easier to test
  - IOT may be cheaper, faster
  - User confidence in system operational capabilities

Protocol Testing George Blankenship 34

The Case for Automated Validation

- If a standard is ratified but not been validated (errors?):
  - Implementations may have errors and affect service operation and assurance
  - Different implementations may have proprietary solutions to overcome errors
    - may not inter-work

Protocol Testing George Blankenship 35

State Explosion Problem

- Even with simple protocols containing a small number of functional entities, messages and states the number of possibilities will be more than most people will have time to verify by hand.
- Some problems have more permutations than a computer can go through, due to limitations on processing speed and memory.

Protocol Testing George Blankenship 36

Travelling salesman problem

- A salesman spends his time visiting n cities (or nodes) cyclically. In one tour he visits each city just once, and finishes up where he started. In what order should he visit them to minimise the distance travelled?
- If every city connected to every other city, number of possible combinations is (n-1)!/ 2. For only 100 cities the number of possible combinations could be as high as 4.67 x 10155
- There is a mathematical proof to show that for the number of towns in the US that the time required to list all combinations would be greater than the estimated length of the universe, and would require more memory than there are atoms in the universe, according to current physics theory.

Protocol Testing George Blankenship 37

Early attempts at Automated Validation

- IBM in Zurich in late ‘70s.
- Perturbation analysis:
  - Starting at a given state
  - Determination of all possible next states
  - Next states as inputs for further perturbations
- Early attempts had drawbacks:
  - No high level formal notation to define protocols
  - Required a large amount of human input
  - Software had to be written for each new protocol

Protocol Testing George Blankenship 38

Formal Design

- **The later** an error is detected **the more expensive** it is to fix.
- Testing the protocol description
- This requires
  - An unambiguous notation
  - Effective validation tools

Protocol Testing George Blankenship 39

ITU Programming Languages

- ITU standardises formal description techniques
  - Z.100 CCITT Specification and description language (SDL)
  - Z.105 SDL combined with ASN.1 (SDL/ASN.1)
  - Z.120 Message sequence chart (MSC)
- ITU defines the following telecommunications languages
  - Z.200 CCITT high level language (CHILL) Common text with ISO/IEC
  - Z.30x Introduction to the CCITT man-machine language

Protocol Testing George Blankenship 40

Performance analysis

- When a new protocol or service is introduced into a network, there needs to be an **understanding what resources are required to implement the service**.
- Performance analysis: WHAT the services resource requirements and quality of service is delivered
- Branch of statistics devoted to understanding communication networks known as ***traffic theory***.

Protocol Testing George Blankenship 41

Traffic Theory

- Well understood for circuit switched, unencrypted voice calls.
- Uses a unit of measurement known as the Erlang.
- 1 Erlang is 1 Hour of calls
- Busy Hour Traffic rate is expressed in Erlangs

(example) If 350 calls are made on a trunk group, and the average call duration is 180 seconds,

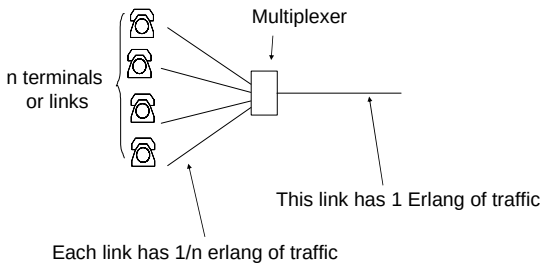
- BHT = Average call duration (s) * Calls per hour / 3600
- BHT = 180 * 350 / 3600
- BHT = 17.5 Erlangs

Protocol Testing George Blankenship 42

Multiplexing

- If every person in the country tried to make a call at the same time, only a portion would get through.
- This is because **lines are multiplexed**.
- Traffic theory enables telecommunication planner to put in enough plant to ensure high probability that every call gets through while minimizing the cost in infrastructure

Multiplexing



Erlang B

- The Erlang B distribution is used for dimensioning trunk routes. It is based on the following assumptions:
- There are an infinite number of sources;
- Calls arrive at random;
- Calls are served in order of arrival;
- Blocked calls are lost; and
- Holding times are exponentially distributed.

Erlang B

- The Erlang B formula is used to predict the probability that a call will be blocked. The Erlang B formula is:

$$B = \frac{\frac{A^N}{N!}}{\sum_{i=0}^{N} \frac{A^i}{i!}}$$

where:
B=Erlang B loss probability
N=Number of trunks in full availability group
A=Traffic offered to group in Erlangs

Protocol Testing George Blankenship 46

Erlang B Example

- I am planning a remote PABX connected by a tie line that will be used for all inbound calls which will have 780 active ends.
- I estimate 30mE of inbound traffic per active end, and GOS should be better than 0.002.
- How many trunks do I need in the tie line route?
- Answer: to carry 780*0.03E=23.4E, I need 38 trunks, and the actual grade of service should be 0.0014

Protocol Testing George Blankenship 47

Erlang C

- The Erlang C distribution is used for dimensioning server pools where requests for service wait on a first in, first out (FIFO) queue until an idle server is available. It is based on the following assumptions:
  - There are an infinite number of sources;
  - Calls are served in order of arrival;
  - Blocked calls are delayed; and
  - Holding times are exponentially distributed.

Protocol Testing George Blankenship 48

Erlang C Formula

- The Erlang C formula is used to predict the probability that a call will be delayed, and can be used to predict the probability that a call will be delayed more than a certain time. From that other key queue performance metrics can be calculated. The Erlang C formula is:

$$P(>0) = \frac{\frac{A^N N}{N!\,(N-A)}}{\sum_{i=0}^{N-1} \frac{A^i}{i!} + \frac{A^N N}{N!\,(N-A)}}$$

where:
P(>0)=Probability of delay greater than zero
N=Number of servers in full availability group
A=Traffic offered to group in Erlangs

Protocol Testing George Blankenship 49

Busy Hour Call Attempts (BHCA)

- Erlang is concerned with occupation of resources
- 1 call of 1 hour duration is same as 6 calls of 10 minutes duration
- For evaluation dynamic performance of modern telecommunications equipment the load on the system of messages flowing trough the system is important
- BHCA is more relevant for evaluating dynamic load

Protocol Testing George Blankenship 50