ABRtest, An Implementation Tool for Conformance Testing the ATM ABR Service Rate Control

David Su, Ernesto Martin-Villalba
High Speed Network Technologies Group
NIST

ABSTRACT

This document describes an implementation of the conformance test suite for the Available Bit Rate (ABR) service rate control scheme that has been proposed in the ATM Forum. This implementation, called ABRtest, is intended to provide a tool for researchers and network designers who wish to construct their own custom test facilities for proving the conformance of their ABR rate control schemes to the proposed standard. The implementation consists of two distinct parts: an Upper Tester (UT) which resides on the same platform as the system under test, and a Lower Tester (LT) which resides on a separate platform. The first part of this document contains the user’s guide for the ABRtest. It begins with an explanation of the architecture of a proposed conformance test system. This is followed by instructions on the operational details the user must know to set up the tests, adjust test parameters, and interpret the results in the test log. The second part gives detailed instructions on how to port ABRtest to the user's system.

1. Introduction

This Implementation Tool, ABRtest, was developed at the National Institute of Standards and Technology (NIST) to facilitate the implementation of the conformance test suite for the Available Bit Rate (ABR) Service rate control scheme specified in section 5.10 of the ATM Forum Traffic Management (TM) Specification V4.0 [1]. The Implementation Tool provides the user with an environment for executing all of the conformance test cases proposed by the ATM Forum [2]. This environment includes a graphical user interface for setting service parameters, setting timer values, and executing the test cases. The test system that is constructed using this tool may then be used for formal conformance testing of an ABR rate control implementation, or for product testing in the early stages of the product development cycle.

The first part of this document is the user's guide for ABRtest. It begins with an explanation of the architecture of a proposed conformance test system. This is followed by instructions on the operational details the user must know to set up the tests, adjust test parameters, and interpret the results in the test log. The second part gives detailed instructions on how to port ABRtest to the user's system.

2. Users' Guide

2.1 System Architecture

Since the proposed rate control scheme specifies end-to-end behaviors of two end systems, the conformance test system must emulate operations of both ends, each end in turn acting as the traffic source and destination. Figure 1 shows the overall system configuration of a testing environment.

The Implementation Tool consists of two major parts: an Upper Tester (UT) which resides on the same platform as the system under test (SUT), and a Lower Tester (LT) which resides on a separate platform, typically an ATM test device. The Upper Tester generates ATM cells through the ATM layer interface of the SUT with the precise timing control needed to force the ABR rate control implementation through its state transitions. The Lower Tester examines the contents of the cells it receives and transmits cells back to the SUT as required. The LT is connected to the SUT through the SUT's physical interface. This ATM connection is established either as a permanent virtual circuit (PVC) or switched virtual circuit (SVC). For synchronization or coordination purposes, the LT and UT need to communicate through a separate connection, either through another VC on the same physical interface, or through an external interface. The software provided utilizes the former approach.

2.2 Design of the Lower Tester

The LT consists of a system independent part and a system dependent part. The system independent part is responsible for the following functions:

- Initialization of parameters
- Coordination with the UT
- Transmission and reception of messages, in ATM cells, in the test cases
- Determination of test verdict, generation of test report

The system dependent part, though the actual implementation may vary, must have at least the following functionality:

- Interface to the ATM device driver for transmission/reception of cells
- Timer implementation: start, reset, expiration

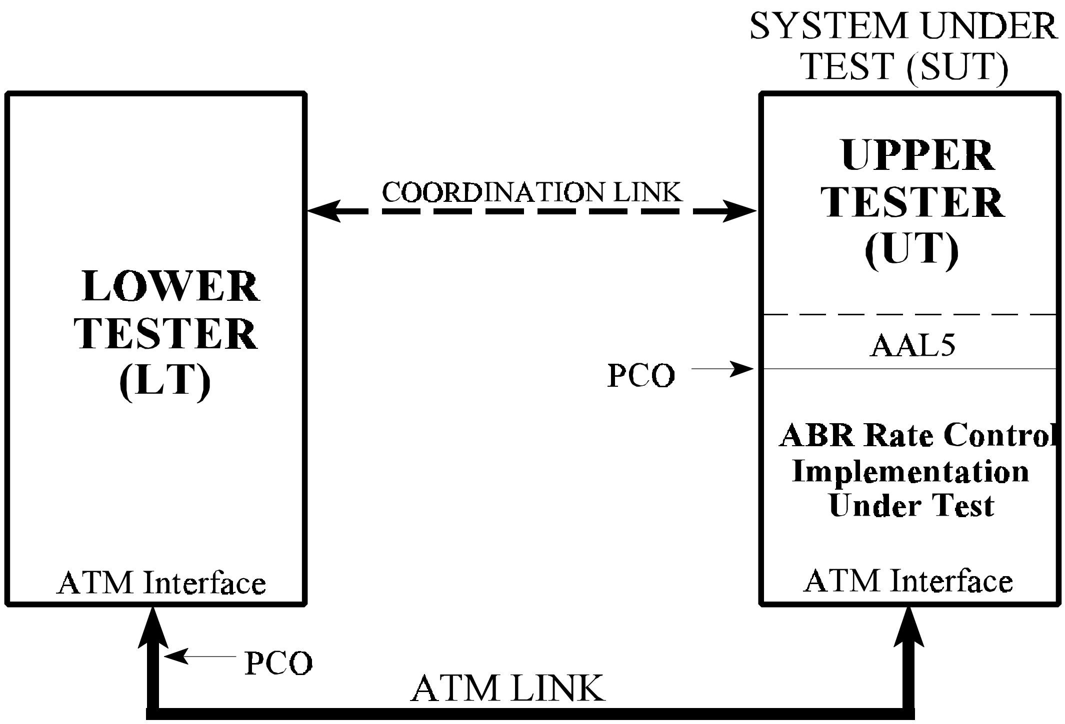

Figure 1. Test System Configuration

The LT transmits and examines ATM cells in and out of the SUT, including Resource Management (RM) cells. Specifically, it examines the contents of the RM cells sent by the SUT to determine if the SUT adjusts its allowed cell rate (ACR) as required. The Point of Control and Observation (PCO) for the cells received/sent by the LT is at the ATM physical interface.

2.3 Design of the Upper Tester

Like the LT, the UT consists of system independent and dependent parts. The system independent part performs the following functions:

- Coordination with the LT
- Transmission and reception of messages, in ATM cells, in the test cases

The system dependent part must perform the following:

- Interface with the AAL5 layer of the SUT
- Establish the ABR connection and the coordination connection between LT and SUT

- Provide multitasking support

The multitasking support is necessary because the UT must be able to simultaneously send/receive messages to/from the LT and SUT.

The Upper Tester transmits ATM cells through the UT PCO to the implementation under test through the ATM Adaption Layer Type 5 (AAL5). It sends one AAL5 Protocol Data Unit at a time to generate one or more ATM cells as required by the test case. If the SUT has an exposed interface at the AAL5 layer, that is there is a programming interface to the AAL 5 layer, then the UT resides above the AAL5 and sends messages to this layer which will then segment the messages into ATM cells. In this case, the UT will send AAL5 Protocol Data of the length such that the exact number of cells (including the AAL5 trailer) are generated. If the SUT has an exposed interface between the ATM and AAL layers, then the UT can be installed to replace the AAL5 and be modified to send ATM cells directly to the ATM layer. The timing and number of cells arriving at the ATM layer are critical factors in the successful execution of the test cases.

2.4 Coordination Between LT and UT

Coordination or synchronization between the LT and UT are done by establishing a virtual circuit between the LT and UT over the ATM link. It is also possible to establish a separate physical link between the tester and SUT, such as a serial data interface. In this case the software must be modified to send/receive from such an interface.

2.5 Operating Procedures

To perform the test, the UT on the SUT must be executed first. Once started, it passively waits for instructions from the LT on the coordination channel, and transmit/receive messages on the ABR VCI as needed. The LT is then started. The LT will perform the following steps in sequence:

- Initialize the ABR rate control parameters
- Establish an ABR connection between the LT and the SUT
- Establish a coordination connection between the LT and the UT
- Execute all test cases that can be performed

At the end, a trace file will contain the results of the test.

2.5.1 Setting the Rate Control Parameters

The procedure for setting the ABR parameters varies, depending on which version of the LT is used. There are two versions of the LT, one with and one without a graphical user interface (GUI) for entering the rate control parameters. With a GUI, the parameter values are typed into labeled boxes on the screen as shown in Figure 2 below. Without a GUI, a predefind configuration file must be created containing the parameter values as shown in Figure 3. The default name of the configuration file is “ A list of the ABR parameters, along with a brief explanation of each, is shown in Table 1.

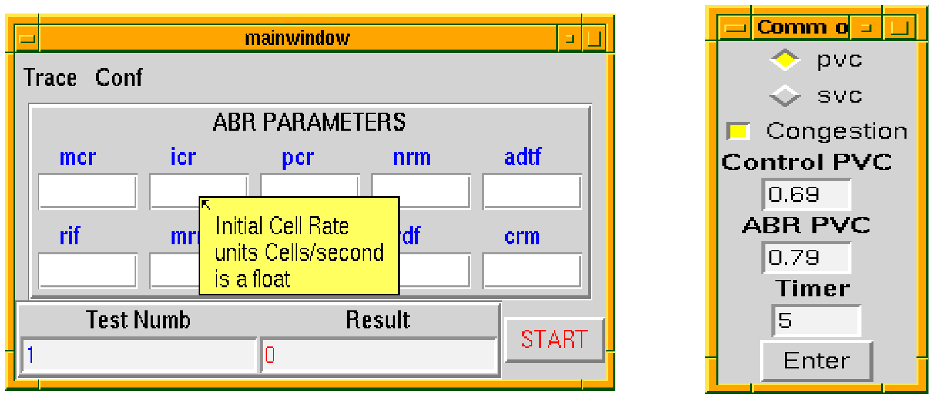

Figure 2. GUI Interface for Entering Rate Control Parameters

Parameter	Symbol	Description	Units	Range
Minimum Cell Rate	MCR	The rate at which the source is always allowed to send.	Cells/sec	Float
Initial Cell Rate	ICR	Rate at which the source should send initially and after an idle period	Cells/sec	Float
Peak Cell Rate	PCR	The cell rate which the source should never exceed	Cells/sec	Float
	Nrm	Maximum number of cells a source may send for each forward RM cell	Power of 2	2 to 256
ACR Decrease Time Factor	ADTF	Time permitted between sending RM cells before rate is decreased to ICR	Seconds	.01 to 10.23
Rate Increase Factor	RIF	Amount by which the transmission rate may increase upon receipt of an RM cell	Power of 2	
Transient Buffer Exposure	TBE	Limit to number of cells sent by source during startup, before first RM cell returns	No. of cells	0 to 16,777,215
	Trm	Upper bound on the time between forward RM cells for an active source	100 ms times a power of 2	100*2e-7 to 100*2e0
Rate Decrease Factor	RDF	Controls decrease in cell transmission rate	Power of 2	
Missing RM Cell Count	CRM	Limits number of forward RM cells which may be sent in the absence of backward RM cells	Integer	

Table 1. ABR Parameters

```
#comment lines should begin with #.
#file for lower tester configuration:
#mcr is the minimum cell rate, the rate at which the source is always
#allowed to send
#units, Cells/second. is a float.
MCR = 250000.0

#icr is the Initial Cell Rate, is the rate at which a source should
#send initially and after and idle period.
#units, Cells/second is a float.
ICR = 950000.0

#pcr is the Peak Cell Rate, is the cell rate which the source may
#never exceed
#units, cells per second, is a float.
PCR = 1100000.0

#Nrm is the maximum number of cells  a source may send for each forward
#RM-cell
#it's a power of 2 from 2 to 256
Nrm = 16

#The ACR Decrease Time Factor is the time permitted between sending
#RM-cells before the rate is decreased to ICR.
#units seconds range from .01 to 10.23 seconds.
ADTF= 5.0

#Rate IncreaseFactor, conrols the amount by which the cell
#transmission rate may increase upon receipt of an RM-cell.
#is a poer of two, the actual rif is 1/2^RIF.
RIF =  2

#Mrm controls allocation of bandwidth between forward RM-cells,
#backward RM-cells, and data cells.
#is constant, fixed at 2.
Mrm = 2

#Transient Buffer Exposure, is the negotiated number of cells that the
#network would like to limit the source to sending during startup
#periods, before the first RM-cell returns
#units cells from 0 to 16,777,215
TBE = 1024

#Trm provides an upper bound on the time between forward RM-cells for
#an active source.
#units, milliseconds its 100 * a power of two.
#from 100*2e-7 to 100*2e0
Trm = 100

#The rate Decrease factor, rdf, controls the decrease in the cell
#transmission rate.
#units as in RIF.
RDF = 02.0

#Missing RM-cell count. limits the number of forward RM-cells which
#may be sent in the absence of received backward RM-cells.
#is an integer.
CRM = 4

#if we can create internal congestion in the NIC.
CONG = 1

#do we use PVCS (1) or SVCS (0)
PVC? = 1

#which is the control PVC?
pvccont = 0.69

#which is the abr PVC?
pvcabr = 0.79

#timer to exit from no return situation.
TIMER = 30

#name of the trace file:
TRACE=suite
```

Figure 3. A Sample Configuration File for Rate Control Parameters

2.5.2 Establishing the ABR Connection

ABRtest does not provide software to establish connections between the LT and SUT. It is the user's responsibility to select the means for making connections. They could be established either manually or automatically through the signaling software available on the LT and SUT. The tool expects the implementor to provide a programming interface routine which, when given the ABR parameters, will establish a connection, and supply the final values of parameters, as well as the virtual path indicator (VPI) and virtual circuit indicator (VCI) used. The current implemention uses a PVC with the following VPI, VCI values which are set in the configuration file as:

VPI = 0
VCI = 69

2.5.3 Establishing the Coordination Connection

The coordination connection could either be a separate VPI/VCI on the same physical ATM link, or other type of external links. The current implementation uses a VBR service PVC with VPI/VCI values set in the configuration file as:

VPIATM = 0
VCIATM = 79

2.5.4 Execution of the Tests and Test Results

Once the ABR connection and the coordination connection have been established, the test cases are automatically executed. A file called **log** is created which contains detailed results of the test, including the time when a cell arrived or was sent, type of cell, and cell contents. An example log file for one test is shown Figure 4 below.

In this example, a command cell was sent to the UT instructing it to send one data cell to the implementation under test. The implementation must send a Forward RM (FRM) cell first before sending the FRM cell. The log file then shows that an RM cell was received by the LT and decodes various significant details of the cell:

HEADER	
ID	Identification (1 byte)
DIR	Direction (1 bit)
BN	Backward Explicit Congestion Notification (1 bit)

CI	Congestion Indication (1 bit)
NI	No Increase (1 bit)
ER	Explicit Rate (2 bytes)
CCR	Current Cell Rate (2 bytes)
MCR	Minimum Cell Rate (2 bytes)

The hexadecimal values for ER, CCR, and MCR are translated into decimal notation. The rest of the cell payload is shown in hex. The last ten bits of the payload are the error code bits.

```
#
Test 1:
Timer: 0.278768
Command cell to UT sent   Send 1 cell(s) to IUT
Timer: 6.24382
Cell received
Resource management cell:
HEADER: 000004f0

ID:     1      DIR:    0      BN:     0      CI:     0      NI:     0
 28     19     ER=     1.09978e+06
 27     a0     CCR=     950272
 23     d1     MCR=     250112
00 00 00 00 00 00 00 00
6a 6a 6a 6a 6a 6a 6a 6a
6a 6a 6a 6a 6a 6a 6a 6a
6a 6a 6a 6a 6a 6a 6a 6a
6a 6a 6a 6a 6a 00 03 e9

Timer: 7.93647
Cell received
HEADER: 000004f0
Payload Type: 0
30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35
36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31
32 33 34 35 36 37 38 39 00 00 00 28 97 38 8f 15
Test Passed
```

Figure 4. A Sample Execution Log File

The next data group in this example shows that a data cell was received; its header and payload are shown in hex notation.

Timer values shown are the elapsed time, in seconds, from the start of the test.

The final line shows the test verdict, which in this case is “Passed”. If the test failed, an indication of why it failed would also be shown.

3. Porting Instructions

The tool is written in C++ and tcl languages. Therefore the development system for porting the tool must have C++ compiler such as gnu and the tcl/tk tools. The tcl/tk tool is needed only for the GUI portion to get the ABR rate control information. The entire system is defined in terms of C++ objects.

The software tool is divided into two major part: the Lower Tester to be executed on some devices capable of generating and receiving ATM cells on a physical interface which is common to this device and the System Under Test (SUT), and the Upper Tester to be executed on the SUT above the ATM and AAL layers.

Each part is further divided into two portions. A system independent part that is independent of the hardware and software platform. This part could be used as is without any change. The other part is for the system dependent functions for which new interface routines must be written on the targeted host systems.

3.1 System Dependent Fuctions

The following is a list of action routines that must be modified by the host system. These are embedded in the related objects.

1. Object `SocketPvc`:

`SocketPvc(Parameters *)` creates the Socket Object using the information provided in the object Parameters. Once a socket is created, it is expected to be able to perform the following funtions:

`recv(unsigned char* data, Atm_hdr *)` returns the ATM cell received for this socket (or the VPI/VCI associated with this socket). The addresses of buffers allocated for the payload and ATM header are provided in `data` and `Atm_hdr` respectively. The first byte of the payload must be stored in byte 0 of the buffer. The first four bytes of the ATM header must be stored as received, the CRC byte is not needed.

`bind (const char *)` binds the socket to the VC indicated in the input. The VC number is in the form VPI.VCI. In the case of PVC, the VC may be binded immediately when the socket is created, and therefore no operation needs to be performed. (For SVC `bind` may initiated the ATM Signaling to make connection.)

`send(unsigned char *, Atm_hdr)` sends an ATM cell to the other side. The first argument points to the buffer of the payload, the second argument contains the first four bytes of the header of the cell. The CRC byte of the header needs to be generated.

`select(struct timeval *timeout)` waits for the arrival of a cell up to the time limit specified in `timeout`. If a cell arrives within time limit, it returns the value TIMEIN indicating a cell is ready for retrieval by the `recv` function. Otherwise, the value TIMEOUT is returned, indicating a time out has occurred.

`close()` closes the connection established on this socket.

A similar object `SocketSvc` creates SVCs and their associated action routines.

2. Object `Timer`:

`timer()` when a `Timer` object is created, the `timer` function is invoked to associate this object with some internal hardware clocks. The logical clock for this object is set to zero and starts to count up with clock ticks in resolution of micro seconds. The clock must also be able to remember a target time which is set and retrieved by the routines described below.

`ellapsed()` returns the time ellapsed in seconds between the creation of this timer object and the current time.

`set(ulong usec)` sets the target time of this timer to the value in `usec` (in unit of micro seconds). The value is specified as time from the creation of the timer.

struct timeval * `remain()` returns a structure timeval with the amount of time remaining since the previous `set` operation, that is the difference between the current time and the target time set previously. The returned value may be negative if the current time has passed the target time.
`wait(ulong usec)` waits for the clock to reach `usec` (in micro seconds since the creation of the timer) before returning control to the caller.

3.2 List of Files in the Package

1. Files for the Lower Tester

System independent part that can be used as is:

```
include/Cell.hh
include/configfi.hh
include/crc.hh
include/crc_10.hh
include/crc_32.hh
include/debug.hh
include/error.hh
include/logfi.hh
include/lower.hh
include/parameters.hh
include/rate.hh
include/tix.h
include/String.h
```

include/Regex.h	
src/Cell.C	contains the object Cell, data structures for ABR RM cells, data cells, and UT command cells.
src/configfi.C	contains objects for reading the information from the configuration file
src/crc.C	for generation of the CRC codes for AAL5 and RM cells.
src/parameters.C	
src/rate.C	translates between the rate format of the RM cells and the native rate format of the machine.
src/lower.c	contains code to execute all test cases and determines verdicts.

The following files are graphical interface for entering parameters, they are needed only if the GUI is used:

include/gui.hh	
include/lowergui.hh	
include/decodergui.hh	
src/gui.C	
src/lowergui.C	
src/decodergui.C	decodes RM, Command and Data cells to obtain the values of various fields.
gui/lower.tcl	
gui/decodergui.tcl	

Files for system dependent part that must be changed:

include/atmdefs.h	
include/timer.hh	
include/network.hh	
src/network.cc	system dependent APIs for sending/receiving atm cells.
src/timer.c	system dependent routing for management of timers

2. There is only one file for the Upper Tester. Both system independent and dependent parts are included in the same file:

src/upper.c	Upper tester on the System Under Test

3. Other files of interest:

src/param	sample configuration file
src/prueba	sample trace file