An AAL5 Performance Test Suite in PerfTTCN

Ina Schieferdecker, Mang Li, Axel Rennoch
GMD Fokus, Hardenbergplatz 2, 10623 Berlin
e-mail: {Schieferdecker, m.Li, Rennoch}@fokus.gmd.de

Abstract

This paper presents a performance test suite for the ATM adaptation layer 5, that can be used to test the end-to-end performance of AAL5 implementations. The test suite includes throughput, frame loss ratio and goodput tests, frame latency tests, and maximum frame burst size tests. It is based on the performance testing white paper of the ATM Forum and shows the usability of PerfTTCN - a performance-extension of TTCN - for the description of performance tests.

1. Introduction

Performance testing is a new methodology to check that a network component (being either a protocol, a service implementation, a telecommunication server or application) meets performance-oriented quality-of-service requirements such as requirements on delays, throughputs, and error rates. Performance testing extends conformance testing [3], which is used to check that an implementation meets its functional requirements and which therefore checks the functional behavior of an implementation only (except of timer dependencies). Performance testing uses additional performance related concepts such as traffic load and measurements.

Different strategies can be used to study the performance of a network component. Either the network component is monitored on-line under real network load or it is tested in an artificial environment using load generators. The first method enables one to study performance under real conditions with a likelihood to monitor unexpected behaviours and that are therefore not addressed in performance tests of the second kind. However, the second method allows precise measurements since the conditions of performance test are completely known and controllable. In particular, the correlations with the measured performance of the network component are less fuzzy than with real network load. Both methods are useful and complement each other.

PerfTTCN [8] aims at addressing both methods. The work was originally motivated by a study of the end-to-end performance of ATM services. Therefore, the approach was based on the ATM Forum performance testing specification [1], which defines the objectives of performance testing for ATM network components and an approach for the development of performance test suites. It intends to assess the quality level of ATM equipment on the basis of performance metrics. Performance metrics are defined for the ATM Layer (i.e. cell error ratio, cell loss ratio, cell misinsertion rate, severely-errored cell block ratio, cell transfer delay, and cell delay variation [1]). In addition, performance metrics for the ATM adaptation layer including lossless, peak and full-load throughput, frame latency, throughput fairness, frame loss ratio, maximum frame burst size, call establishment latency, and application goodput are defined. Different connection configurations are proposed for performance tests.

PerfTTCN is an extension of TTCN [4], what is the only standardized, well known and widely used notation for the description of conformance tests. PerfTTCN uses additional constructs for the definition of performance tests such as the declaration of load models, network configurations, performance characteristics, and performance measurements as well as the control of measurements, load generators, and monitors.

This paper presents a performance test suite for AAL5 that shows the usability of PerfTTCN for the description of industrial relevant performance tests. In particular, the results of describing performance tests are promising: the performance test suite is succinct and understandable and reduces the risk of misinterpretations.

The paper is structured as follows. The basic concepts of performance testing are introduced first before presenting a short overview of PerfTTCN. The main part is constituted by the AAL5 performance test suite itself.

2. Introduction to performance testing

The goal of performance testing is to check the performance of an implementation under normal and overload situations. The conditions under which a performance test is executed are described uniquely in order to reduce the possibilities of misinterpretations and to make test results repeatable and comparable to other test results. The test conditions are described via the configuration of the implementation under test (IUT), the configuration of the network, and the load characteristics.

We assume that an IUT is conformance tested before any performance test is applied. That ensures that the test results are not impacted by an incorrect behavior of the IUT. However, since overload may degrade the functional behaviour, care has to be taken when the test results are analyzed. PerfTTCN is based on the following concepts:

- A performance test consists of several, possibly distributed foreground and background test components, which are coordinated by a main tester.
- A foreground test component realizes the communication with the IUT. The foreground tester uses discrete test events to bring the IUT into specific states in which the performance measurements are executed. The foreground tester may generate a continuous stream of data packets to emulate the foreground load for the IUT.
- A background test component generates continuous streams of data and loads the network. It does not directly communicate with the IUT. It only implicitly influences the IUT as it brings the network and the IUT into minimal, normal, or overload situations.
- Traffic models describe continuous streams of data. For example, one could use Markov Modulated Poison Processes (MMPPs) for the description of traffic loads [5].
- Points of Control and Observation (PCOs) are the access points for the foreground and background test components to interfaces of the IUT or of the network. PCOs are used to send and receive data as well as to collect time stamps of test events for performance measurements.

To sum up, a performance tests uses an ensemble of foreground and background tester and points of control and observation. Traffic models define the generated load. The performance test configuration describes the conditions under which the measurements are executed. It is the description of the performance test configuration that makes performance test experiments re-usable and performance test measurements comparable.

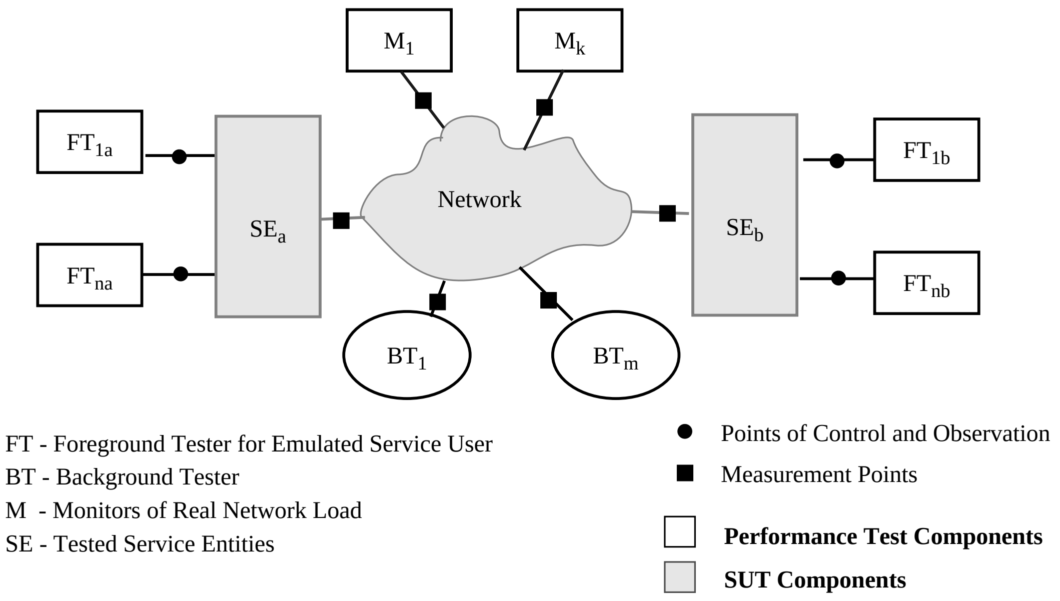
M_1
M_k
FT_{1a}
FT_{na}
SE_a
Network
SE_b
FT_{1b}
FT_{nb}
BT_1
BT_m
FT - Foreground Tester for Emulated Service User
BT - Background Tester
M - Monitors of Real Network Load
SE - Tested Service Entities
Points of Control and Observation
Measurement Points
Performance Test Components
SUT Components

Fig. 1 Performance test configuration for an end-to-end service

In [8] three different performance test configuration for performance tests of an application, of an end-to-end transmission service, and of a protocol has been defined. Only the configuration of a performance test for an end-to-end transmission service (see also Fig. 1) is of interest to the presented test suite. The System Under Test (SUT) consists of the IUT and network components. The test components are not shadowed. For simplification, the main tester is not included in this figure. The performance test uses foreground tester at both ends of the end-to-end service, which emulate the service user. Background tester are used to generate artificial load to the network. Measurements points (a specific type of PCOs) are used to monitor the real load in the network during the performance test.

A measurement collects time stamps of test events. A measurement is started once and continues until it is explicitly cancelled or reaches its end. Based on the measurements, more elaborated performance characteristics such as mean, standard deviation, maximum and minimum as well as the distribution functions can be evaluated.

The evaluation uses predefined performance metrics. The evaluation of performance characteristics is supported either by an on-line analysis (i.e. during test execution) or by an off-line analysis (i.e. after the test finishes and all samples have been collected).

The main tester uses control constructs to start and cancel background test components and measurements and to access the actual measurements (via performance constraints). It assigns test verdicts that do not only assess the observed behaviour of the IUT to be `passed` or `failed`, but also returns the measured performance characteristics as part of the test report.

3. The PerfTTCN Approach

The performance test configuration (Table 1) defines the background traffic components and the source and destination of the generated load. The coordination points are used to control the traffic generation and monitoring.

Test Component Configuration Declaration			
Configuration name: CONFIG_2			
Components Used	**PCOs Used**	**CPs Used**	**Comments**
MTC	PCO_1	CP1,	
...			
Background Test Component			
Identifier	**PCOs Used**	**CPs Used**	**Comments**
traffic1	(PCO_B1) -> (PCO_B2)	BCP1, BCP2	Point to Point
...			

Table 1 Integration of Background Test Components

The generated background traffic itself is defined by traffic models. An example for a constant bit rate traffic is given in Table 2.

Traffic Model Declaration	
Name: const **Type**: CBR **Comments**:	
PCR	10 MBit/s

Table 2 CBR Traffic Model Declaration

The purpose of the background traffic is to create load on a network that will be transmitted through the system under test. Specific details on e.g. connection information such as VPI/VCI for an ATM connection are subject to the PICS or PIXIT documents[3]. Each background traffic may select number of instances of traffic models. Each of these traffic streams is identified by a name that can be used in the dynamic behaviour part to start the corresponding traffic load. A background traffic stream that uses the model of Table 2 is given in Table 3.

Background Traffic Stream Declaration			
Traffic Name	**Background Test Component**	**Model Name**	**Number of Instances**
Load1	traffic_1	const	6
...			

Table 3 Background Traffic Stream Declaration

A performance measurement (Table 4) is declared by a metric and combined with two test events (and related constraints) to indicate the start and termination of the measurement. A set of standard metrics like counter, delay, jitter, frequency, throughput with a well-defined semantics is used. For example, `delay_filo` is the delay with a first bit send and last bit arrived semantics[1]. User defined metrics (implemented with test suite operations) can also be used.

Measurement Declaration							
Name	**Metric**	**Unit**	**event 1**	**constraint 1**	**event 2**	**constraint 2**	**Comments**
res_delay	delay_filo	ms	PCO_1 !Request	s_req_spc	PCO_2 ?Response	r_resp_spc	

Table 4 Measurement declaration

Measurements are the basis to evaluate characteristics such as means, frequency distributions, maximum, minimums, etc. These are defined in a performance characteristics declaration (see also Table 5). In order to be statistically significant, a measurement needs to be repeated several times, so that a sample size or a duration can be declared.

Performance Characteristics Declaration					
Name	**Calculation**	**Measurement**	**Sample size**	**Duration**	**Comments**
res_delay_mean	MEAN	response_delay	20		

Table 5 Performance characteristics declaration

PerfTTCN distinguishes between functional constraints based on PDU and ASP value matching, which are the traditional constraints in TTCN, and measurement related constraints called performacne constraint.

Performance Constraint Declaration		
Name	**Constraint Value Expression**	**Comments**
p_resp	(res_delay_mean < 5) AND (res_delay_max < 10)	

Table 6 Performance constraint declaration

A performance constraint can be used for the on-line assessment of the observed performance, so that the performance test can be finished whenever the performance goes beyond a given limit. The performance constraint declaration (Table 6) consists of a name and a logical combination of expressions, where an expression consists of performance characteristics with individual thresholds.

The control of the traffic loads and performance measurements is specified in the behaviour descriptions of test cases or test steps. They have to be started explicitly and may be cancelled. Performance constraints are indicated in the constraint reference column.

Due to lack of space we refrain from giving an example here and refer to Table 9.

4. The Performance Test Suite for AAL5

The performance test suite `AAL5_CP_Perf` for performance tests of the end-to-end transmission service of AAL5-CP (ATM Adaptation Layer Type 5 Common Part) is presented in this section.

AAL5 is specified by the ITU-T in I.363 [5] and defines three sublayers for AAL5: the Segmentation and Reassembly sublayer (SAR) and the Common Part Convergence sublayer (CPCS) build the AAL5-CP and offer the common functionalities of AAL5 to a user. The Service Specific Convergence Sublayer (SSCS) may be null

1. In general, four different semantics can be given to a delay measurement: FILO = first bit in, last bit out, FIFO = first bit in, first bit out, LIFO = last bit in, first bit out, and LILO = last bit in, last bit out.

for AAL5, and is therefore not considered.

The main function of AAL5-CP is a non-assured data transfer with error detection and handling. The maximum size of an AAL5 packet are 65535 octets. AAL5-CP uses the CPCS service primitives: CPCS-UNITDATA-invoke and CPCS-UNITDATA-signal. The performance metrics at frame level [1] are used to define the performance of an AAL5 transmission service between two AAL5-CP users. The performance metrics include metrics for throughput, frame loss ratio and goodput, frame latency, and maximum frame burst size.

The performance test suite uses abstract service primitives (ASPs) in accordance with the CPCS specification to send and receive AAL5 data to and from the implementation under test (IUT). The occurrences of the ASPs are also the dedicated events that are observed by the performance measurements and are therefore the basis to derive performance characteristics and to assess the performance of the IUT.

The performance test suite uses the distributed test method and assumes that an API at the upper service boundary of the IUT exists which supports the access to the CPCS abstract service primitives.

4.1 Performance Test Configuration

The performance test configuration of an AAL5_CP performance test is based on the generic configuration for an end-to-end transmission service that was introduced in Section 3. Fig. 2 presents the test configuration of AAL5_CP_Perf. It makes use of multiple parallel test components[2] to control and monitor artificially generated traffics as well as to monitor real network load.

The System Under Test (SUT) consists of two ATM end systems ESa and ESb and of one ATM switch SW that connects both end systems. The IUT as part of the SUT is indicated by the shadowed area.

The test system includes the main test components MTC that controls the general flow of the test behavior, and several local test components:

- foreground test components to control and observe foreground traffic (FTxa, FTxb),
- sources and destinations of background traffic (BTxa, BTxb, BTxc), and
- emulation parts to support BTxc on end systems that do not belong to the IUT (EM),

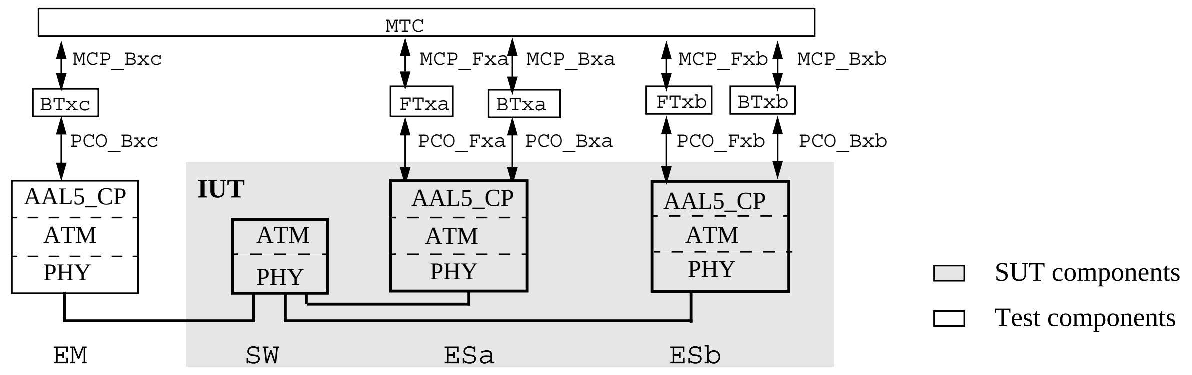

Fig. 2 Test Configuration of AAL5_CP_Perf

In addition, the test system uses coordination points between the local and the main test component (MCP_*) and points of control and observation to communicate directly with the IUT (PCO_*).

This configuration in Fig. 2 uses only a minimal number of test and system components. However, the test configuration of Fig. 2 is generic. Each of the local test components supports a specific functionality for the performance tests, so that multiple instance of the local test components with “x” ranging from 1 to n may be used in a concrete performance test (provided that available hardware allows to do so). The test suite has been specifically aimed at supporting scalability by means of a modular configuration.

The scalability of the test configuration is of particular use when bringing the IUT into overload situations. The IUT can be either overloaded at the end systems or at the switch[3]:

- an end system is loaded via the upper service boundary of AAL-CP using BTxa as the source and BTxb as the

2. Therefore, the additional proformas and mechanisms of Concurrent TTCN [4] are used in the performance test suite.
3. Please note that for example in ATM LANs, the end systems are usually the bottlenecks for data and protocol processing, while switches provide larger capacities for data transmission.

destination of background traffic (or vice versa.)

- a switch is loaded via an upper service boundary of AAL-CP of one or more endsystems that do not belong to the IUT. `BTxc` is used as the source of the background traffic while either another `BTxc` or `BTxa`/`BTxb`[4] are used as the destination. EM supports the communication with the switch.

The concrete configuration of a performance test has to be defined in related documents: the PICS (Protocol Implementation Conformance Statement) and/or the PIXIT (Protocol Implementation eXtra Information for Testing) documents have to be used here. Only a unique description of the concrete performance test makes the test results and SUTs comparable.

4.2 Test Suite Structure

The test suite `AAL5_CP_Perf` is structured into test groups, which correspond directly to the connection configurations introduced in [2] (see also Fig. 3).

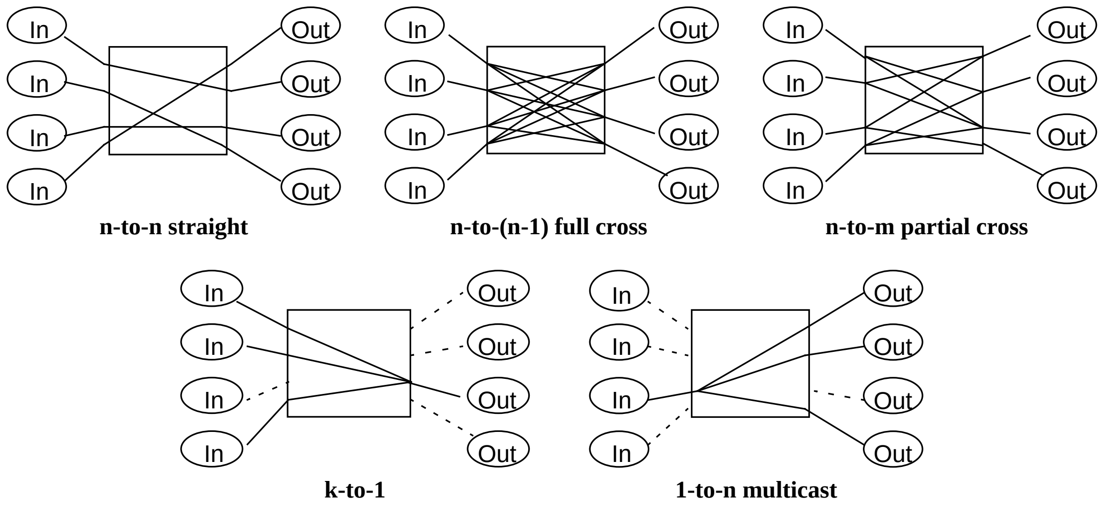

Fig. 3 Connection Configurations

The current version of the test suite contains only test cases for the one-to-one-straight configuration with no background traffic[5]. Therefore, the test system is restricted to `MTC`, `FT1a`, `FT1b` and the connecting `MCPs` and `PCOs`. The following test cases[6] are included:

- `TP001` for Throughput, Frame Loss Ratio and Goodput tests,
- `FL001` for Frame Latency tests, and
- `BS001` for Maximum Frame Burst Size (MFBS) tests.

Each test case uses an `MTC` to start the foreground test components with the corresponding behaviours that are described in their test steps. During the test execution, the `MTC` communicates with foreground test components to control their test activities. Two basic functions of the foreground test components are specified by the test steps:

- `FTxaSendFrame_y` defines the behaviour of `FTxa` to send frames and to perform measurements at the sending side, and
- `FTxbRcvFrame_y` defines the behaviour of `FTxb` to receive frames and to perform measurements at the receiving side.

The concrete behaviour of an `FT` depends on the test case in which it is invoked. “`y`”, which equals `TP`, `FL` or `BS`, is used to distinguish between different behavior. `FTxa` and `FTxb` are either determined to serve as the source or the destination of the foreground traffic, respectively. In order to use the opposite direction of data flow and measurement, the inverse mapping of the logical names to the physical system components is needed only.

4. This offers means to overload an end system indirectly via the lower service boundary.
5. This restriction was mainly induced by the ongoing work at the ATM Forum to define additional connection configurations and to define guidelines for the use of performance metrics in different configurations.
6. The test suite does not contain throughput fairness tests since only one VC is considered. In addition, call establishment latency tests do not belong to a AAL5_CP performance test, since AAL does not deal with any signalling procedures.

4.3 Frame Latency Tests

The test case for frame latency tests (FL001) is used to describe the performance test behavior in more detail. ATM Forum[2] defines a message-in message-out (MIMO) semantics for the frame latency performance metrics[7]. We selected the last-in last-out (LILO) definition to simplify the test: LILO is the delay between the last-bit entry and the last-bit exit. Under the precondition that the input rate at the receiver of the IUT is less or equal the output rate at the sender of it, MIMO equals LILO. This precondition is checked in a question of the PICS document in order to determine the Selection Reference InputSEOutput (meaning input rate small or equal output rate) to be true or false.

According to [2], the recommended test procedure for frame latency tests is as follows:

- a sequence of equally spaced frames is sent with a rate of lossless throughput for a predefined period with duration TTL (total time of frame latency test),
- some of the frames in the flow are marked, and
- the LILO latency of the marked frames are recorded.
- Finally, the mean value and standard deviation are computed from the samples.

The test procedure in FL001 reflects the collection of samples while the statistic values are computed off-line by the use of performance evaluation procedures. The foreground test components and the IUT exchange test frames that are of ASP type CPCS_UNITDATA_inv and CPCS_UNITDATA_sig [5]. The test frames use two special fields in the payload to support the performance tests:

- four octets are reserved for a sequence number that is used to identify each test frame uniquely and
- one octet is reserved to mark special test frame for the Frame Latency test.

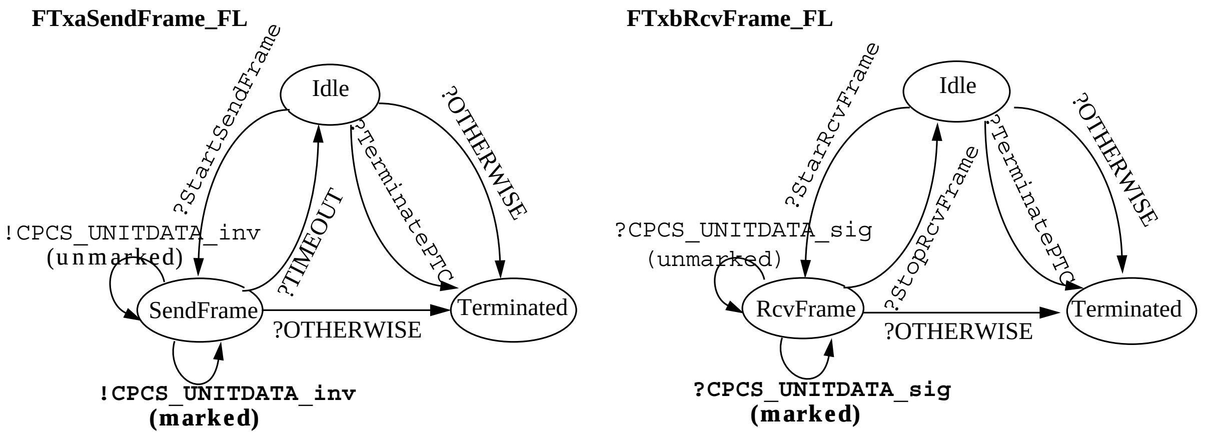

Fig. 4 State Machines of the Foreground Test Components

The coordination messages CMs that are exchanged between the MTC and the foreground test components are in pairs similar to a send and reply scheme (except of SendFrameStopped and Error):

• StartSendFrame:	MTC sends FTxa this message to request the start of sending activities. It includes two parameters: four octets for the initial sequence number and one structured field to convey some traffic model related information such as frame input rate and frame burst size.
• SendFrameStarted:	FTxa acknowledge MTC the StartSendFrame message and enters the SendFrame state.
• StartRcvFrame:	MTC sends FTxb this message to request the start of receiving activities.
• RcvFrameStarted:	FTxb acknowledges MTC the StartRcvFrame message and enters the RcvFrame state.
• SendFrameStopped:	FTxa sends MTC this message to notify that the time of one experiment expires and that the sending activities are stopped, it enters the Idle state.
• StopRcvFrame:	MTC sends FTxb this message to stop the receiving activities.
• RcvFrameStopped:	FTxb acknowledges MTC the StopRcvFrame message and enters the Idle state.

7. Please refer to [1] and [2] for further details of the MIMO specification.

• `TerminatePTC`:	`MTC` sends `FTxa` or `FTxb` this message to terminated it.
• `PTCTerminated`:	`FTxa` or `FTxb` acknowledge `MTC` the `TerminatePTC` message and enter the `Terminated` state.
• `Error`:	`FTxa` or `FTxb` sends `MTC` this message whenever an unexpected event occurs.

Test Step Dynamic Behaviour					
Test Step Name:	FTxbRcvFrame_FL (mcpFxb: CP; pcoFxb: AAL5CP_UT)				
Group:	TestBody/RcvFrame/				
Objective:	Used in Frame latency test.				
Default:					
Comments:					
Nr	**Label**	**Behaviour Description**	**Constraints Ref**	**Verdict**	**Comments**
1	Lxb01	mcpFxb?TerminatePTC	TermPTC_sr1		
2		mcpFxb!PTCTerminated	PTCTerm_sr1	R	PTC is terminated.
3		mcpFxb?StartRcvFrame	StartRF_sr1		
4		mcpFxb!RcvFrameStarted	RFStarted_sr1		
5	Lxb02	pcoFxb?CPCS_UNITDATA_sig	Frame_Vr1TP		unmarked frame is rcvd.
6		GOTO Lxb02			
7		pcoFxb?CPCS_UNITDATA_sig	Frame_Vr2FL		marked frame is rcvd.
8		GOTO Lxb02			
9		mcpFxb?StopRcvFrame	StopRF_sr1	(P)	End of experiment.
10		mcpFxb!RcvFrameStopped	RFStopped_sr1		
11		GOTO Lxb01			
12		?OTHERWISE			
13		mcpFxb!Error	Er_sr1	(F)	
14		?OTHERWISE			
15		mcpFxb!Error	Er_sr1	(F)	
Detailed Comments:					

Table 7 Foreground Test Component FTxbRcvFrame_FL

The behavior of `FT1a` and `FT1b` is represented in form of state machines[8] in Fig. 4. The initial state of `FTxaSendFrame_FL`, when `FT1a` is started by `MTC` using the operation `CREATE`, is `Idle`. `FT1a` enters the next state `SendFrame` when the `StartSendFrame` message is received. Test frames, unmarked or marked, are sent to the IUT until the predefined test duration expires. Subsequent to that, `FT1a` enters `Idle` again and waits for the next `StartSendFrame`. If in the `Idle` state `TerminatePTC` is received instead, `FT1a` enters the `Terminated` state and the test step finishes. Error cases are taken into account (by means of `?OTHERWISE` statements) in order to avoid deadlocks. `FTxbRcvFrame_FL` as the receiving entity shows the opposite behavior to `FTxaSendFrame_FL`. Unmarked or marked frames are received from the IUT in the state `RcvFrame` until a `StopRcvFrame` message is received. Table 7 contains the description of `FTxbRcvFrame_FL` in TTCN.

The measurement for marked frames in a frame latency test is defined in Table 8. A sample of the measurement has to be taken whenever `FT1a` receives a marked frame (see also Table 7, line 7). Table 8 defines an identifier `frameLatency` for this measurement. The identifier enables `MTC` to access the measurement and to start and stop it. `Event_1` with `constraint_1` denotes the sending of a marked frame, while event_2 with constraint_2 indicates the receipt of a marked frame[9]. The measurement uses the predefined metric `LILO_DELAY`, that is to measure the delay between the last bit in of `event_1` and the last bit out of `event_2`. `FrameLatency` is a distributed measurement, it starts at `PCO_F1a` (with `event_1`) and ends at `PCO_F1b` (with `event_2`). It measures the delay between LI of a frame of type `CPCS_UNITDATA_inv` with content

8. For the ease of understanding, the state machines in •Fig. 4 are simplified by omitting the `MCP_Fxa`, `PCO_Fxa` and the outgoing messages to `MTC`.
9. Please note, that event_2 with constraint_2 corresponds to Line 7 in Table 8. That indicates that the measurement takes a sample whenever a marked frame is received.

Frame_Vs2FL(seqNum) that leaves PCO_F1a and enters the IUT, and LO of a frame of type CPCS_UNITDATA_sig with content Frame_Vr1FL that exits the IUT and arrives at PCO_F1b.

Measurement Declaration						
Name	**Metric**	**event 1**	**constraint 1**	**event 2**	**constraint 2**	**Comments**
frameLatency	LILO_ DELAY	PCO_F1a!CPCS_ UNITDATA_inv	Frame_Vs2FL(seq Num)	PCO_F1b?CPCS_ UNITDATA_sig	Frame_Vr2FL	Frame latency of marked frames.

Table 8 Measurement Declaration

Finally, let us consider how the FrameLatency measurement is controlled by the MTC (test case FL001 in Table 9). New PerfTTCN language constructs are used in line 5 and 14 only. At first, the ATM virtual connection that is used by the performance test is initiated (line 1). Afterwards, the test components FT1a and FT1b are created (line 3, 4). At line 5, MTC starts the frameLatency measurement via the extended START construct. Coordination messages are exchanged between MTC and FT1a and between MTC and FT1b to coordinate the begin and end of a test experiment (line 6-13). A test experiment ends when frameLatency is terminated (using the extended CANCEL construct, line 14). If additional test experiments are needed, e.g. in Maximum Frame Burst Size tests, the cycle between START and CANCEL of the measurement may be repeated.

Otherwise, before the test finishes, FT1a and FT1b are terminated and the data connection is released (by means of the postamble, line 15). Finally, the statistics of the measured frame latency are calculated (line 16). The two timers TDly and TTL are used to avoid deadlocks, whenever either the communication between MTC and FT1a or FT1b is disturbed or the FT1a or FT1b are not able to respond to MTC requests for any reason.

5. Conclusions

The paper presents a performance test suite for ATM Adaptation Layer 5. It has been written in the new language proposal of the TTCN performance extension PerfTTCN. With PerfTTCN test suite developers are able to use the known TTCN constructs as well as new language elements for expressing performance characteristics, measurements, and other conepts of performance testing. This approach opens an easy means to specify new requirements which are upcoming with performance testing.

The frame latency test is shown only, however the performance tests for throughput, frame loss ratio and goodput as well as for the maximum frame burst size are written in a similar manner. The current version of the performance test suite contains one-to-one straight configuration tests. These will serve as a basis to develop test cases for the other connection configurations. In addition, we intend to extend the performance tests with background traffic components and to include on-line performance calculations such as whenever the frame latency exceeds a predefined maximum during the performance test, it is stopped immediately.

The development of a runtime environment for PerfTTCN is under work. As soon as it is available, the presented performance test suite will be applied to ATM adapter cards (e.g. from ForeSystems or Sun). The results of the performance tests will be compared with measurement results of their end-to-end performance.

References

[1] ATM Forum: ATM Forum Traffic Management Specification, af-tm-0056.000, April 1996.

[2] G. Babic, A. Durresi, R.Jain, J.Dolske, S. Shahpurwala: Proposed modifications to Performance Testing Baseline: Throughput and Latency Metrics, ATM_Forum/97-0426, Apr. 1997.

[3] ISO/IEC 9646-1 “Information Technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 1: General Concepts“, 1991.

[4] ISO/IEC 9646-3, “Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 3: The Tree and Tabular Combined Notation (TTCN)”, Delivery 8.4, Oct. 1996.

[5] ITU-T I.363 - B-ISDN ATM Adaptation Layer (AAL) Specification, March 1993.

[6] R. Jain, et al: ATM Forum Performance Testing Specification - White Paper of the ATM Forum. - BTD-TEST-TM-PERF.00.01 (96-0810R4), Jan. 1997.

[7] R. O. Onvural: Asynchronous Transfer Mode Networks: Performance Issues. Artech House Inc., 1994.

[8] I. Schieferdecker, B. Stepien, A. Rennoch: PerfTTCN - a TTCN language extension for performance testing. - accepted to appear at the 10th Intern. Workshop on Testing of Communicating Systems (IWTCS’97), Cheju Island, Korea, Sept. 1997.

[9] Telelogic: ITEX 3.1 User Manual, Sweden, Dec. 1996.

Test Case Dynamic Behaviour					
Test Case Name:	FL001				
Group:	NToNStraight/OneToOne/WithoutBgrTraffic/				
Purpose:	Frame Latency performance test for one-to-one straight configuration without background traffic.				
Configuration	Straight11_WithoutBT				
Default:	DEF1				
Comments:					
Selection Ref:	InputSEOutput				
Description:	Frame Latency performance test for one VC without background traffic.				
Nr	**Label**	**Behaviour Description**	**Constraints Ref**	**Verdict**	**Comments**
1		+Init_Straight11_WithoutBT			Initialize the VC.
2		(inputF:=InputFIni_FL)			
3		CREATE (FT1b: FTxbRcvFrame_FL(MCP_F1b, PCO_F1b))			Start the receiving PTC.
4		CREATE (FT1a: FTxaSendFrame_FL(MCP_F1a, PCO_F1a))			Start the sending PTC.
5		START frameLatency			Start measurement.
6		MCP_F1b!StartRcvFrame START TDly	StartRF_sr1		Start receiving activities
7		MCP_F1b?RcvFrameStarted CANCEL TDly	RFStarted_sr1		
8		MCP_F1a!StartSendFrame START TDly	StartSF_s2FL(inputF, iniNum)		Start sending activities
9		MCP_F1a?SendFrameStarted CANCEL TDly	SFStarted_sr1		
10		START TTL (TTL_Value + TDly_Value)			Start Timer TTL.
11		MCP_F1a?SendFrameStopped CANCEL TTL	SFStopped_sr1		Sending activities are stopped.
12		MCP_F1b!StopRcvFrame START TDly	StopM_sr1		Stop receiving activities
13		MCP_F1b?RcvFrameStopped CANCEL TDly	RFStopped_sr1		
14		CANCEL frameLatency		(P)	Terminate measurement
15		+pa_Straight11			Postamble.
16		[calStatistics_FL() =TRUE]		R	Calculate statistics.
17		?TIMEOUT TTL		(I)	The value of TTL could be too small.
18		+pa_Straight11			Postamble.
Detailed Comments:					

Table 9 Frame Latency Test `FL001`