TTCN-3 based Implementation of ETSI TISPAN IMS Benchmark

George Din, Diana Vega, Razvan Petre, Andreas Hoffman

Fraunhofer Institute for Open Communication Systems

Contents

- Motivation
- What do we test?
- How do we test?
- Implementation details
- Benchmark Results
- Conclusions
Motivation

- Goal - performance benchmark for IMS components
  - Performance and scalability testing of all IMS and related components with simulated real-world traffic
  - Measurement and analysis of important QoS parameters

- Why
  - Creation of objective means to compare overall IMS of different systems by performance (and price)
  - Check ability of hardware/software to run the IMS

- How
  - Define standard scenarios and traffic models for the work load
  - Define the metrics to be measured
  - Standardize the test procedure, the test parameters and the Benchmark test report

- Where
  - Standardization of IMS benchmarking at ETSI TISPAN WG6
  - Version 1.0 of IMS benchmarking standard has been released.
Benchmark Information Model

Use case 1 (e.g.: registration)

Scenario 1.1
(e.g. user not registered)
- Message flow
- Design objectives
- Metrics & graphs to collect
- SuT parameters

Scenario 1.2
(e.g. already registered)

Use case 2 (e.g.: session setup)

Scenario 2.1
(e.g. successful call)
- Message flow
- Design objectives
- Metrics & graphs to collect
- SuT parameters

Scenario 2.2
(e.g. rejected call)

Benchmark Test

Preamble Registration Phase

Traffic Set
- X% of scenario 1.1
- Y% of scenario 2.1
- Z% of scenario 2.2

Traffic Profile
- Initial load
- Step duration
- Step Increase Amount
- Number Of Steps
- Other parameters

Test Report

Disclaimer

Benchmark Test Run
- SuT configuration and parameters
- Test System configuration and parameters
- Metrics and graphics with Design Objective Capacity (overall and for each scenario)

Observations and interpretations

Exceptions

Scenario: Successful Call

Registration

Client 1 IMS CN
- REGISTER 401
- REGISTER 200 OK

Client 2 IMS CN
- REGISTER 401
- REGISTER 200 OK

Section Tear Up Tear Down

Client 1 IMS CN
- INVITE
- 100 Trying
- 180 Ringing
- Ringing Time
- 200 OK
- ACK
- ACK

Client 2 IMS CN
- REGISTER
- 200 OK

CN
- INVITE
- 180 Ringing
- Ringing Time
- 200 OK
- ACK

Client 1
- BYE
- 200 OK

Client 2
- BYE
- 200 OK

Talking Time
Scenario: Successful Call

Section Tear Up Tear Down

Client1      INVITE      INVITE
            100 Trying    180 Ringing
            180 Ringing
            200 OK

Client2      ACK
            ACK

CN

BYE

TRT-SES1

TRT-SES2

TRT-REL1

Talking Time

Ringing Time

TTCN-3 User Conference, 30.05.2007

Benchmark Procedure

Estimated Maximum Load

Initial Call Load

Load Ramp

Rate Increase

Load Steady

Stability Or Setting Time

Stop Condition Met

Time To Stop Condition

Fluctuations, Steady state?

Linear, Immediate?

Stability?
User Concept

1. A user is a state machine running a scenario
2. A user may be "callee" or/and "caller"
3. A user may create > 1 calls
4. A user may be reused to create other calls
5. A user may call randomly any other user

Simultaneous Scenarios

- Same user involved in more than another instance of *same* scenario
- Same user involved in *different* scenario
Population Clusters

- **registration** (not registered yet)
- **deregistration** (registered and not involved in any other scenario)
- **all other scenarios like messaging, calling, reregistration**

TTCN-3 Message Types

```
// general type for a SIP Request
// can be extended by any other request type (i.e. INVITE_Request)
type record Request {
  integer transactionId,
  RequestLine requestLine,
  MessageHeader msgHeader,
  charstring messageBody optional
}

type set MessageHeader {
  Authorization authorization optional,
  CallId callId optional,
  Contact contact optional,
  CSeq cSeq optional,
  Expires expires optional,
  From fromField optional,
  RecordRoute recordRoute optional,
  Route route optional,
  ServiceRoute serviceroute optional,
  To toField optional,
  Via via optional,
  MaxForwards maxForwards optional,
  ContentLength contentLength optional,
  WWWAuthenticate wwwAuthenticate optional
}

// [20.7 RFC2617 3.2.2]
type record Authorization {
  FieldName fieldName(AUTHORIZATION_E),
  // Credentials body
  charstring body optional
}
```
TTCN-3 State Handling

alt ( // P-CSCF --------REQUEST--------> User
  (p2SUT.receive (TRequest) -> value request { // P-CSCF --------INVITE--------> User
    if (match(request, Request_INVITE_r)) {
      // P-CSCF --------MESSAGE-------> User
      else if (match(request, Request_MESSAGE_r)) {
        // P-CSCF --------ACK--------> User
        else if (match(request, Request_ACK_r)) {
          ..............
        }
      }
    }
    // P-CSCF --------RESPONSE--------> User
    (p2SUT.receive (TResponse) -> value response {
      // P-CSCF -------- 200 OK --------> User
      if (match(response, Response_200_r)) {
        if (response.msgHeader.cSeq.seqMethod == "INVITE") {}
        else if (response.msgHeader.cSeq.seqMethod == "BYE") {}
        else if (response.msgHeader.cSeq.seqMethod == "REGISTER") {}
      }
      // P-CSCF -------- 401 OK --------> User
      else if (match(response, Response_401_r)) {
        ..............
      }
    }
  }
)

Messages Creation

// P-CSCF --------RESPONSE--------> User
[] p2SUT.receive (TResponse) -> value response {
  // P-CSCF -------- 200 OK --------> User
  if (match(response, Response_200_r)) {
    if (response.msgHeader.cSeq.seqMethod == "INVITE") {
      ackReq := ACK_Request_s;
      ackReq.transactionId := getNewTransactionId();
      ackReq.requestLine.method := ACK_E;
      ..............
      ackReq.msgHeader.callId := response.msgHeader.callId;
      ackReq.msgHeader.cSeq := response.msgHeader.cSeq;
      ackReq.msgHeader.cSeq.seqMethod := "ACK";
      ..............
      ackReq.msgHeader.fromField := response.msgHeader.fromField;
      ackReq.msgHeader.toField := response.msgHeader.toField;
      ackReq.msgHeader.route := getServiceRoute(response);
      userChannel := getChannel(response);
      p2SUT.send (ackReq) to userChannel;
      delTransactionId(response.transactionId);
    }
  }
}
Processing of Messages

Platform adaptor

transaction state

message processor

identify
validate
update state
process response

REQ
RESP
SUT

test component

Adaptation Layer

Aim: Maximize TTCN-3 system performance!

- Handles multiple parallel connections with SUT
- Uses Pools of Threads and Listeners
- Smaller number of
  - threads and context switches
  - mutex-es and semaphores
  - contended message queues

Test System

Test Management (TM)

Component Handling (CH)

TTCN-3 Executable (TE)

Coding & Decoding (CD)

System Adapter (SA)

Platform Adapter (PA)

System under Test
Advantages:

- I/O operations are made completely separately by the processing of the message.
- Concurrent I/O are possible on a SMP system.
Distribution of SAPS

Average Delay for Invite Requests
Conclusions

- TTCN-3 specification has a big impact on the performance of the hole test system
  - Using a component to simulate more than one user
  - Using a component sender
  - Using a tree like mechanism for matching the messages

- Efficient Adaptation Layer Design has an important role in the overall performance
  - Smart codec/decoder system
  - Non-blocking I/O for protocol stack
  - Optimized runtime for specific needs of the benchmark

Thank you!

Time for questions ...