|  |
| --- |
|  |
|  |
| **S1 Simulator**  **USER GUIDE**  **Version 1.0** |
|  |
|  |

**Contents**

[1 Preface 3](#_Toc482018116)

[2 Solution overview 4](#_Toc482018117)

[2.1 Introduction 4](#_Toc482018118)

[2.2 Architecture: 4](#_Toc482018119)

[2.3 S1 Simulator Modules: 5](#_Toc482018120)

[3 S1 Simulator directory structure 7](#_Toc482018121)

[3.1 Test Controller Application 8](#_Toc482018122)

[3.2 Test Controller Stub 9](#_Toc482018123)

[3.3 Traffic Generator 9](#_Toc482018124)

[4 Compilation of S1Simulator modules 10](#_Toc482018125)

[4.1 Compile S1SIM Application 10](#_Toc482018126)

[4.2 Compile Traffic Generator 10](#_Toc482018127)

[4.3 Compile Test Controller Stub 10](#_Toc482018128)

[5 API Definitions 11](#_Toc482018129)

[5.1 APIs for Test Controller 11](#_Toc482018130)

[5.2 APIs for Traffic Generator 11](#_Toc482018131)

[6 Configuration and Execution 12](#_Toc482018132)

[7 Traffic Testing 15](#_Toc482018133)

[7.1 Test Scenario Executed: 15](#_Toc482018134)

[7.2 Traffic testing on stand-alone machines 15](#_Toc482018135)

[7.3 Traffic testing on Virtual Box setup 16](#_Toc482018136)

[8 Limitations: 18](#_Toc482018137)

# Preface

The document provides the compilation and execution steps of S1 Simulator components.

Author assumes that the readers of this document are:

* Product Development team,
* Test or Validation team,
* Program Management team,
* Customers

The readers must have an understanding of LTE technology and eNodeB functionality.

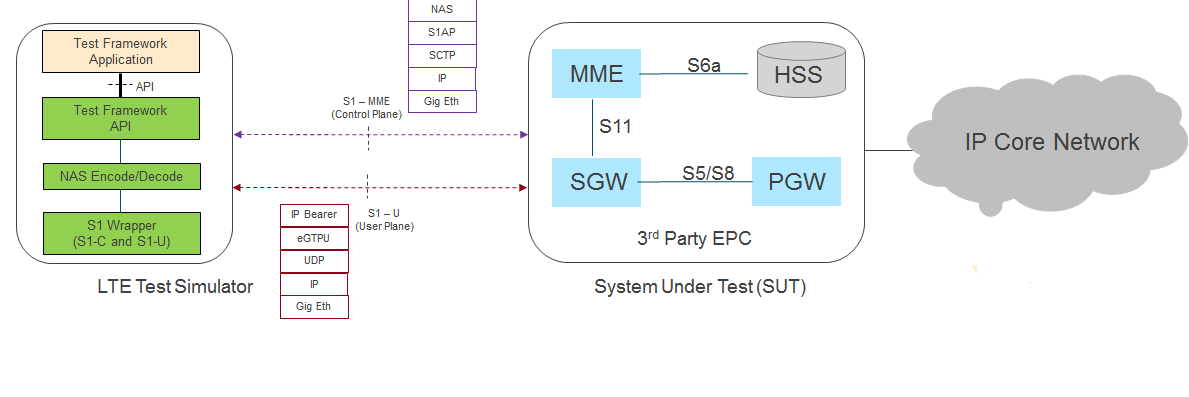
# Solution overview

## Introduction

* S1-Simulator simulates UE(s) and eNodeB in LTE Access Network
* The simulator is used for verifying the EPC S1-C and S1-U Interface.
* S1-Simulator provides hooks for generating Control and user plane procedures on the S1 Interface.
* S1-Similator can be used to test the core functionalities and to measure the performance of a EPC in LTE

## Architecture:

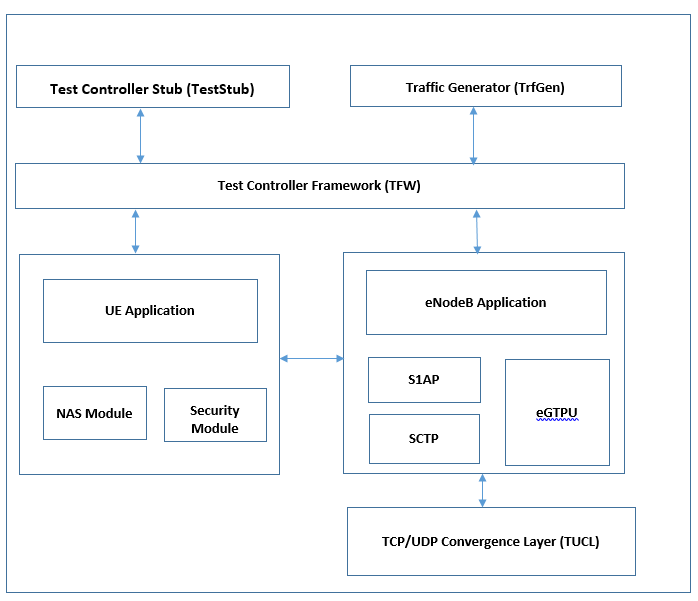
The Network Architecture of S1- Simulator is shown below:



**Fig-1: Network Architecture of S1 Simulator**

## S1 Simulator Modules:

LTE Simulator /S1 Simulator solution has modules as shown in the below Fig-2:



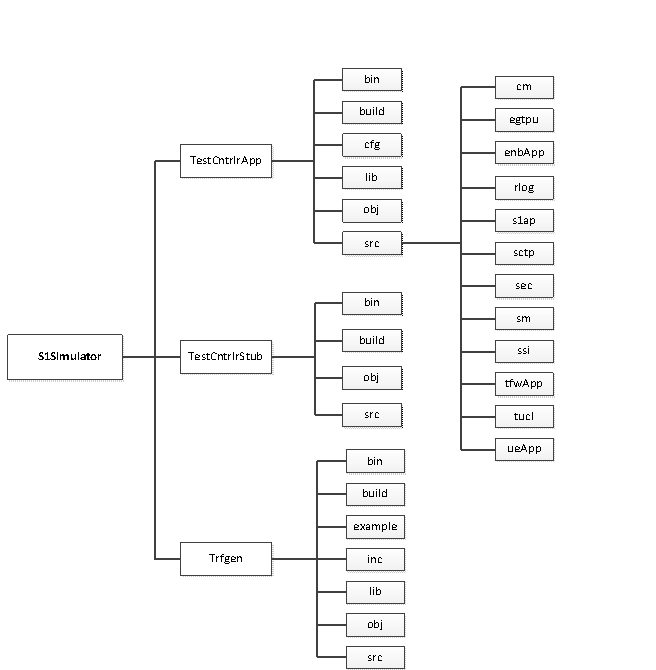
**Fig -2: S1 Simulator Modules**

The S1 Simulator product contains the following components:

* **Test Controller Framework (TFW)**
  + Exposes APIs towards the Test Controller Stub
  + Responsible for configuring and interacting with UE Application and EnodeB Application layers
* **UE Application (ueApp)**
  + Simulates the UE
  + Maintains UE related configuration and call processing data. Also has state machines per UE
  + Has NAS codec and security modules to build the NAS messages
* **EnodeB Application (enbApp)**
  + Simulates the EnodeB functionality
  + Maintains S1 Connection details
  + Has S1AP and eGTP protocol stacks.
  + Has TUCL Convergence layer, implements TCP and UDP layers
  + Responsible for setting up of S1 Control and User plane
* **Traffic Generator (Trfgen)**
  + Provides API for trigerring User plane traffic over LTE bearers of a simulated UE.
  + It uses Iperf3 for traffic generation
* **Test Controller Stub (TestStub)**
  + Contains example test cases which use the APIs provided by test framework for simulating various control and user plane procedures.
  + Triggers various test case events towards Test Controller Framework (TFW)
  + Receives and processes events from TFW

# S1 Simulator directory structure

Following figure lists the directory structure of S1 Simulator:



**Fig-3 : S1 Simulator Directory Structure**

## Test Controller Application

Following table lists the sub-directories and contents of TestCntrlrApp:

|  |  |  |
| --- | --- | --- |
| **Directory Name** | **Sub-Directory Name** | **Description** |
| bin |  | Test Controller Application Library |
| build |  | Makefiles for building the Test Controller Application |
| cfg |  | Default configuration files |
| lib |  | Other product libraries |
| obj |  | Object files |
| src | cm | Common files, interface definition files |
| egtpu | Egtpu stack component files |
| enbApp | eNodeB Application |
| rlog | Logging framework |
| s1ap | S1AP stack component files |
| sctp | SCTP stack component files |
| sec | Security Module |
| sm | Stack Manager Modules |
| ssi | System Services Interface |
| tfwApp | Test Framework applications |
| timer | Timer framework files |
| tucl | Implementation of TCP/UDP stack(convergence layer) |
| ueApp | UE Application |

## Test Controller Stub

Following table lists the contents of TestCntrlrStub:

|  |  |
| --- | --- |
| **Directory Name** | **Description** |
| bin | Test Controller Stub Executable |
| build | Makefiles for building the Test Controller Stub |
| obj | Object files |
| src | Source code files |

## Traffic Generator

Following table lists the contents of Trfgen:

|  |  |
| --- | --- |
| **Directory Name** | **Description** |
| bin | Traffic Generator Library |
| build | Makefiles for building the Traffic Generator |
| example | Example files to use Traffic Generator |
| inc | Header files |
| lib | Library |
| obj | Object files |
| src | Source code files |

# Compilation of S1Simulator modules

S1Simulator binary can be generated by executing below compilation steps:

## Compile S1SIM Application

1. Go to “build” directory of Test-Controller-Application

$ cd TestCntlrApp/build

1. Compile the S1 Simulator source code.

$ make cleanall

$ make

On successful compilation, the “libtfw.so” library gets generated under **TestCntlrApp/lib** folder.

## Compile Traffic Generator

1. Go to “build” directory of Traffic generator

$ cd Trfgen/build

1. Compile the Traffic generator source code

$ make clean

$ make

On successful compilation, the “libtrfgen.so” library gets generated under **Trfgen/lib** folder.

## Compile Test Controller Stub

1. Go to “build” directory of Test-Controller-Stub

$ cd TestCntlrStub/build

1. Compile the S1 Simulator Stub-code

$ make clean

$ make

On successful compilation, the **“testCntrlr”** binary gets generated under **TestCntrlStub/bin** folder.

**Note**: All three binaries compiled on Ubantu-14.04 .

# API Definitions

## APIs for Test Controller

Attached spreadsheet contains various APIs for Test Controller:



## APIs for Traffic Generator

Following table lists the APIs for Traffic Generator:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Primitive | Direction | Description | Parameters | Message id (enum defined in trfgen.x file) | Structure name (defined in trfgen.x file) |
| trf\_test\_init | TestStub -> Trfgen | Initialization of Traffic generator. Does configuration of client and servers | None | NA | NA |
| startDlData | TestStub -> Trfgen | Starts DL Traffic for an attached UE | Bind IP Address(UE IP) | NA | NA |
| startUlData | TestStub -> Trfgen | Starts UL Traffic for an attached UE | Bind IP Address(UE IP) | NA | NA |

# Configuration and Execution

Before executing S1Simulator binary all configurable parameters must be set in respective configuration files.

All configuration files for S1Simulator are available under TestCntlrStub/bin folder.

1. **imsi.txt**

Contains list of IMSIs for the simulated UEs

Example:

imsi.txt

|  |
| --- |
| 2 0 8 1 1 1 2 3 4 5 6 7 8 9 0 /\* UE-1 imsi \*/  5 0 5 0 2 9 8 7 6 5 0 0 5 0 6 /\* UE-2 imsi \*/  5 0 5 0 2 9 8 7 6 5 0 0 5 0 9 /\* UE-3 imsi \*/ |

1. **nbAppCfg.txt**

Contains eNodeB related configuration parameters

Example:

nbAppCfg.txt

|  |
| --- |
| CELL\_ID 1 /\* cell ID \*/  TAC 1 /\* Tracking Area Code \*/  ENB\_IP\_ADDR 192.168.200.62 /\* EndoeB IP Address \*/  MME\_ID 1 /\* mme ID \*/  MME\_ADDR 192.168.200.160 /\* MME IP Address \*/  SCTP\_IP\_ADDR 192.168.200.62 /\* SCTP IP Address \*/  ENB\_NAME "XYZ" /\* EnodeB Name \*/  PLMN\_ID 50502 /\* PLMN Supported \*/  HEARTBEAT\_INTERVAL 5000 /\* SCTP Heartbeat Interval(MilliSecs) \*/  RTO\_INITIAL 200 /\* SCTP: RTO INITIAL \*/  RTO\_MIN 100 /\* SCTP: RTO MINIMUM \*/  RTO\_MAX 1000 /\* SCTP: RTO MAXIMUM \*/  S1\_PREP\_TIMER\_VAL 5000 /\* S1 Preparation Timer Value (MilliSec) \*/  S1\_OVERALL\_TIMER\_VAL 10000 /\* S1 Overall Timer Value (MilliSecs) \*/  SCTP\_UDP\_SERVICE\_TYPE 0 /\* UDP Service Type \*/  INACTIVITY\_TIMER\_VAL 2000000 /\* Inactivity Timer Value (MilliSecs) \*/  MAX\_EXPIRY 200 /\* Maximum Expiry count for Inactivity Timer \*/  S1\_SETUP\_TMR\_VAL 2000 /\* S1 Setup Timer Value(MilliSecs) \*/  NO\_OF\_SCTP\_IN\_STREAMS 3 /\* SCTP Incoming Streams Nos. \*/  NO\_OF\_SCTP\_OUT\_STREAMS 3 /\* SCTP Outgoing Streams Nos. \*/  UE\_ETH\_INTF eth0 /\* Interface on Which Ue Configured \*/  VS\_IP\_ADDR 192.168.200.4 /\* Video Server IP Address \*/  CFGEND |

1. **ueAppCfg.txt**

Contains ueApp related configuration parameters

Example:

ueAppCfg.txt

|  |
| --- |
| TRF\_GEN\_IP\_ADDR 172.0.0.0 /\* Traffic Generator IP Address \*/  AUTH\_TYPE XOR /\* Authentication Type \*/  NAS\_CYPH\_CFG eea0 /\* NAS Ciphering configuration \*/  NAS\_INT\_PROT\_CFG eia1 /\* NAS Integratory Configuration \*/  NAS\_PROC\_GUARD\_TIMER 10 /\* Guard Timer Value for NAS messages \*/  NUM\_OF\_UE 1 /\* No of Ue Supported \*/  CFGEND |

1. **ueCfg.txt**

Contains Common UE configurations

Example:

|  |
| --- |
| ALGO\_TYPE AES /\* security algorithm supported \*/  …  ….  EEA0 TRUE /\* Is this EPS encryption algorithm supported \*/  …  AUTH\_TYPE MILENAGE /\* Authentication type \*/  …  SHARED\_KEY 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 /\* shared key \*/  …  …  CFGEND |

S1Similator can be run executed by following below steps.

1. Configure and bring up all EPC modules.
2. Go to bin directory of Test-Controller-Stub and update following configuration parameters

$ cd TestCntlrStub/bin/

* + Update valid IMSIs in imsi.txt.
  + Update ENB IP address, MME IP address, ENB name, PLMN and timer values in nbAppCfg.txt.
  + Update Authentication, Ciphering and Integrity types in ueAppCfg.txt.
  + Update OP KEY and SHARED KEY in ueCfg.txt.

1. Export LD\_LIBRARY\_PATH:

$ export LD\_LIBRARY\_PATH=../../TestCntlrApp/lib/:../../Trfgen/lib/

1. Help for executing S1Similator displayed by running below command. It displays are supported test cases.

$ testCntrlr -h

1. Running non-UE related test cases (S1 SETUP, eNB Config etc):

$ testCntrlr <test\_case\_keyword>

Example 1:

$ testCntrlr S1SetupSucc

1. Running UE related test cases (Attach, Detach, TAU etc):

$ testCntrlr <test\_case\_keyword> <no\_of\_UEs>

Where test\_case\_keyword tells which test case to run and no\_of\_UEs tells how many UEs to simulate.

Example 1: to test end to end attach with 1 UE

$ ./testCntrlr EndToEndAttach 1

Example 2: to step by step attach with 5 UEs

$ ./testCntrlr StepAttachWithImsi 5

1. Running a list of test cases given in a text file:

$ testCntrlr –f <file\_name>

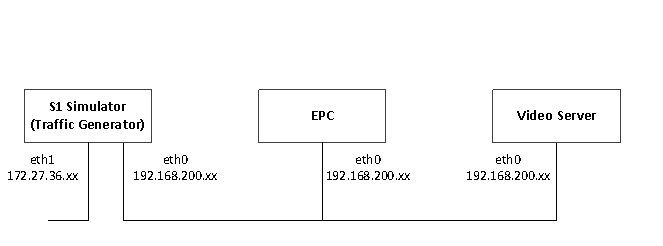
Example 1:

$ testCntrlr –f testCaseList\_1.txt

# Traffic Testing

## Traffic testing on stand-alone machines

The S1 Simulator communicates with EPC on Ethernet port eth1. Create UE IP addresses by binding to Ethernet port eth0 of S1 Simulator machine. Refer the below network diagram for test setup.



**Fig-4: Data Test Setup**

To test the Uplink data:

1. Configure the Video server IP address in ts\_utls.x

In ts\_utls.x a macro is defined with name the VS\_IP\_ADDRESS, change the video server IP address here.

1. Start the iperf servers binding to port numbers [5001, 5002, … 5032] on Video server.
2. Create virtual IPs for UEs.
3. Run the binaries with command line option AttachAndUlData.

./testCntrlr AttachAndUlData <number of Ues>

To test the Downlink data:

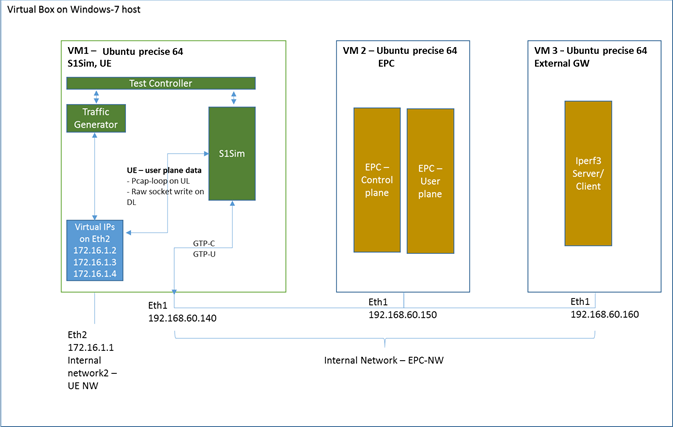
1. Configure the Video server IP address in ts\_utls.x

In ts\_utls.x a macro is defined with name the VS\_IP\_ADDRESS, change the video server IP address here.

1. Start the iperf clients binding to port numbers [7001, 7002, … 6032] on Video server.
2. Create virtual IPs for UEs.
3. Run the binaries with command line option AttachAndDlData.
4. ./testCntrlr AttachAndDlData <number of UEs>

## Traffic testing on Virtual Box setup

The following Test setup is used for verifying User-plane data.



**Fig-5: Data Test Setup on Virtual Box Setup**

VM1 :

* + S1Sim runs pcap\_loop to capture Uplink UE data plane packets generated from Traffic generator from *UE-Traffic-Intf*. The Interface name for *UE-traffic-Intf* is configurable in *nbAppcfg.txt*.
  + Virtual IPs are created **manually** for all UE IP addresses that shall be assigned by EPC
  + Upon successful Attach and bearer establishment, Test controller starts traffic for the UE by invoking the APIs of Traffic generator. Traffic generator uses libiperf3 to generate traffic.
  + Uplink traffic is captured by pcap-loop and mapped to corresponding GTP channel based on the source IP address in the generated packets(UE IP address assigned by PDN)
  + Downlink traffic received on GTP tunnel is forwarded to the UE by writing to the *UE-Traffic-Intf* using raw socket.

Other configurations used on VM1:

* Route all packets with Desination IP as external GW to eth2(*UE-Traffic-Intf)*
  + - * *Eg: IP route add 192.168.60.160 device eth2*
    - Disable ARP on eth2(*UE-Traffic-Intf)*
      * *Ifconfig eth2 -arp*

# Limitations:

* The S1-simulatror validated for maximum 32 ues attach and data(uplink/downlink)
* Maximum throughputs observed with S1Sumulator and traffic generator running on a single X86 desktop with Ubuntu Linux Ver 14.04 is given below.

|  |  |  |  |
| --- | --- | --- | --- |
| **No. of Ues** | **TCP/UDP** | **Direction** | **Throughput** |
| 32 | TCP | Uplink | 1 Mbps/UE |
| 32 | TCP | Downlink | 400 Kbps/UE |
| 32 | UDP | Uplink | 400 Kbps/UE |
| 32 | UDP | Downlink | 400 Kbps/UE |