Research Paper Engineering

A Tool for Performance Testing of User Plane Sub-System of Radio Network Controller (3G)

Prithvi Shankar N	Digital Communication Engineering, RVCE, Bengaluru, India.
Sisil John	Nokia R&D Technology Centre, Bengaluru, India.
Prakash Goyal	Nokia R&D Technology Centre, Bengaluru, India.
RK Manjunath	Digital Communication Engineering, RVCE, Bengaluru, India.

ABSTRACT

In present day scenario, the numbers of mobile phone users are growing day by day but, the systems deployed in the telecom network will not be changed nor replaced as the number of users grow for the same Technology (Ex 2G or 3G network). Therefore the systems which are deployed in the customer environment should be capable of handling the extreme scenarios like overload conditions to the normal scenarios with optimal load. If the system has to handle all the scenarios without any unexpected events or unit or system level crash, then all the systems have to undergo thorough testing for its Performance characteristics and capture all the defects in the functionality in the test environment.

Radio network controller (RNC) is an important system in 3G networks. It does some of the major functionalities like Radio resource allocation, ciphering, Service area broadcast etc. There are many sub systems in RNC, which handles each of the work performed by RNC, out of which User Plane sub system is the one which handles the actual user data that flows through the network, be it Voice or data. Therefore since the number of mobile users has direct impact on the User plane sub system, the measurement of Performance Characteristics of it is a mandatory step.

The Performance testing of User plane sub system involves lot of repetitive processes like triggering the number of call to test each scenario and also the test runs usually extend for long durations due to which manual intervention to check the results would be tedious and sometime prone to human errors. This initiated the requirement to automate the procedure of performance testing of User plane. Hence this paper aims at implementation of a tool for performance testing of USER PLANE sub system, using Perl as the Scripting Language with help of call simulators and test data tools which will reduce the testing run time as well as reduce the human errors that tend to occur during manual testing.

KEYWORDS UMTS; 3G; SDLC; O&M; SUT;

I. Introduction:

3G is one of the widely used mobile communication technologies for data transfer at high speeds. The basic UMTS (Universal Mobile telecommunication system) network consists of BTS/nodeB (Base station), RNC (Radio network controller), Core network and the link between them is as shown in Fig1.

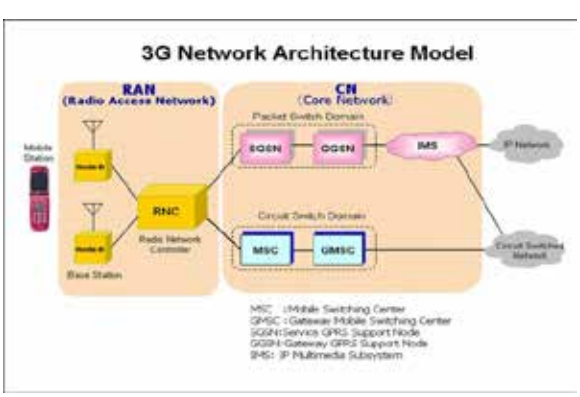

Fig1: 3G network architecture

RNC plays an important role in the UMTS network. The important functionalities of RNC are Radio resource management and mobility management is also done to limited extent. Radio resource management includes controlling of transmitter power, data rates, handover, modulation type and channel coding parameters etc.

The major sub-components in RNC are Control Plane, User Plane and Operation and Management (O&M).Control plane processes only signaling data whereas User Plane is the one which handles the actual User Data messages.

Testing is an important step in the SDLC (Software development life cycle)which is an important process in software domain. Any product before releasing it to the customer has to be tested end to end considering all valid scenarios. Testing can be wholly divided into white box and black box testing. In white box testing the tester would know the details of the software or the product as such, whereas in black-box testing testers would not know the system behavior. They need to give different input to the system, and check the output, based on the output they have to come to a conclusion on the characteristics of the system.

In the white box testing there are many sub types, but in this paper we concentrate only on Performance testing. The reason is during performance testing, real environment would be created and System under test (SUT) is subjected to load which is seen in already commercially deployed system. The system under test considered here is the User plane component of the Radio Network controller.

Performance testing is a tedious task. It involves lot of tools, configurations to be done, bringing up the environment and triggering the load on the SUT and then verifying the characteristics needed to validate the performance parameters. Since it involves lot of manual work and testing is done for more than a day, manual intervention to debugging into any failure of the run would be a time consuming and repetitive task. Hence an automated tool needs to be developed, which reduces the manual load and also makes the task easier. Therefore the scope of this paper is to develop an automated tool

to perform the performance testing of the User plane component of Radio network controller system.

II. Performance Testing
There are sub classifications in the performance testing, based on the amount of load applied on the system.

a. Load testing:
In this method, SUT is subjected to normal load. The amount of load applied would be decided after the analysis of the load on the system deployed in customer environment.

b. Stress Testing:
This type of testing will determine the robustness of the system. The SUT will be tested at maximum load for long durations to check the behavior under extreme conditions. Fig 2 gives the scenario for stress testing.

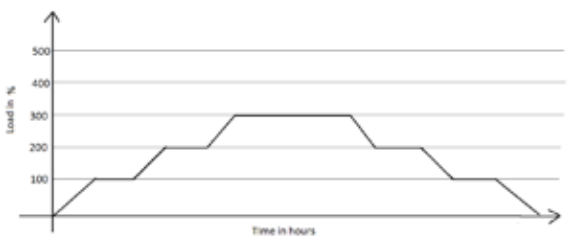

Fig 2: Load testing graph

c. Spike Testing:
This testing is achieved by varying the load within a short span of time and observing the behavior. It can determine whether SUT can sustain dramatic changes in the load.(Ex during Festi-

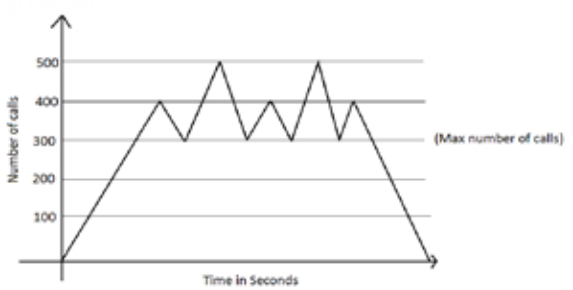

Fig 3: Stress testing Graph
Hence to certify that performance characteristics of SUT are stable, it has to be tested under all the 3 scenarios mentioned.

III. Implementation of tool:
Here the SUT is considered as User Plane component of RNC system. To create the actual load in a lab environment, simulators are needed which can pump in the load to the SUT. Hence Nokia proprietary tools are used which include simulators to help in simulating the call traffic and another tool to hold the test cases and test data which will serve as inputs during the automated run.

The overall functionality of the tool is depicted in flow chart below.

The entire implementation shown in above flowchart can be divided into 3 major steps,

1. Simulator handling
2. Log collection
3. Log analysis

1. Simulator Handling:
This step will determine the number of calls to be pumped into the SUT. This is handled by giving input to the simulator as to how many number of UE need to be triggered.

There are 2 input files for this step, one is Config.ini and another is .ODS file.

Firstly the config.ini file contains, as to how many UE's and nodeB's have to be initialized in the simulator. The initialized number should always be greater than what needs to be triggered. The config.ini will be given by manual testers based on the test case.

Second is the .ODS which is as shown in Fig4.

Fig4: .ODS file

It contains, First column as group number, Second column as number of UE per group, Third Column as Inter group delay and intra group delay (only for overload conditions).

This number of UE's per group is decided upon the load i.e. generated on the deployed RNC's. They calculate the total number of call hits per second to the RNC deployed in customer environment, during normal and busy hour and finally decide on the number of UE's. The inter group delay is amount of delay which is introduced between groups. The main aim here is to create load on the SUT as shown in the Fig 2, the num of calls grows steadily, remains high during the busy hour and towards end of the day goes down.

The group number will be mapped to service type inside the simulator. This mapping will be taken care in the config.ini file. For Ex Group 1 will be mapped to AMR call type which is basic voice call and group 2 is mapped to HSPA which includes both voice as well as data. By doing these combinations we can recreate the actual load on the SUT.

The call duration parameter is mentioned in the Test tool. It is given by the manual testers. Since it is performance testing the call durations will always be in many hours and sometimes 2 days over the weekend.

The script written for simulator handling will do the following functionalities.

1. Copy the config.ini and .ODS file from testers machine to the Simulator machine
2. Open the simulator GUI and load the Config.ini file
3. Then trigger the nodeB's, the number of nodeB's required are taken as input from tester
4. The UE's are triggered group wise, taking care of inter/intra group delay from the group 0 to the last in a sequential manner. The script will read the .ODS file line by line and send the trigger command with the required group id and number of UE's
5. Then running all the calls based on the Call duration parameter taken as input from tester
6. Then the UE's are stopped in the reverse order as mentioned in .ODS file, with delay taken care.
7. Then nodeB's are stopped and finally Simulator GUI is closed.

2. Log collection:
In this step, logs collected are the ones by which the performance characteristics of SUT can be measured. The logs are Memstat (memory statistics)

Clogs (Computer logs)
Proc Stat (Process Statistics)
Bit Rate logs
Alarms
Drop statistics.

Memstat: These logs are collected to check for memory leaks issue. Each process is allocated memory and these processes are run on DSP cards. Hence memory statistics of all DSP cards are collected and only required memory pool which is for User plane is analyzed. It is collected before and after the call.

Procstat: These logs are collected to monitor the CPU utilization. It is collected per second throughout the call duration.

Computer logs:
These logs give the detail about the exceptions that would have occurred during the call process. There are error codes which occur in the logs, based on which analysis is done.

Bitrate logs:
This log gives the total data which gets generated during the call, both in uplink and downlink i.e. in the Iu-b and Iu interfaces.

Drop statistics:
This log gives the packets lost during the call at service level i.e. the services which are running in the DSP cards.

Alarms:
It records any alarms that occur during the call process. The general examples for alarms are disconnection between units, system crash etc. Based on the description the severity of the alarm can be defined and analysis is done based on the severity.

The steps followed to collect logs are

1. Login to RNC machine and inside that login to individual DSP cards
2. Execute the log collection commands
3. Redirect the command output to a text file with proper naming convention
4. Then copy the file back to the windows machine where analysis can be done.

The above steps are done using modular scripts for each of the logs which are collected. These scripts are called whenever required, i.e. before the call has begun (before simulator handling), during the call and after the call. At the end of the log collection step, the scripts written to collect the logs are integrated with Simulator handling step and then verified that all the mentioned logs are collected.

3. Log Analysis:
Analysis of the logs is done on the windows machine i.e. the logs collected on RNC are copied to the local machine and then Analysis Scripts are triggered.

1. Memstat Analysis.
Firstly we check for memory statistics. The log is as shown in fig5.

```
Memory statistics
-----------------
Pool                0
Size/Requested/Allocated/Maximum 1044480 /   52424 /   90016 /   90080 bytes
```

Fig 5: Memstat logs
Only a part of log is shown in Fig5. Once the Memstat is collected for all DSP cards, we analyze only the memory of pool 4, which is allocated to user plane process. Here the allocated bytes(the highlighted field in Fig5) field is measured for before and after the call. If there are any differences found then we can conclude that a process is still running which is consuming memory.

This can create serious problems when the SUT is tested under overload or maximum load condition. In such scenarios when memory is not released, the used memory builds up and finally there won't be memory available for new calls processes, which eventually leads to call drops.

The analysis output is as shown below

memstat_DMPS_0_dsp_0	BEFORE CALL	AFTER CALL	memstat_DMPS_0_dsp_1	BEFORE CALL	AFTER CALL
POOL0	90320	90320	POOL0	141176	HASH=[0x7442c1f0] FAIL
POOL1	12288	12288	POOL1	75288	
POOL2	90112	90112	POOL2	163168	
POOL3	16904	14964	POOL3	53176	
POOL4	2572744	2572744	POOL4	141168	
POOL5	0	0	POOL5	0	
POOL6	0	0	POOL6	0	
POOL7	0	0	POOL7	0	

Fig 6: Analysis output of Memstat
From the above fig by observing the highlighted blocks we can conclude that the memory allocated is same before and after call for pool4 in first DSP0 but in the DSP1 the memory statistics is not collected, due to which comparison fails and verdict is set as FAIL. Hence the final Analysis would depend on Memstat analysis of all the DSP cards.

2. Clogs Analysis:
Clogs give the exceptions that occur during the call process. The fig 7 gives the positive and Fig 8 gives negative scenario. In positive scenario as seen in fig 6, no error code has appeared.

Fig7: Positive Scenario of Clogs

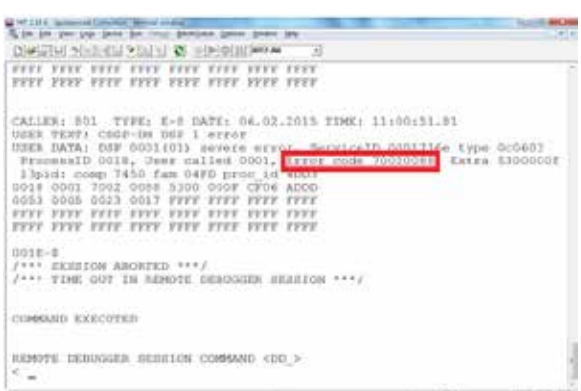

Fig 8: Negative Scenario of Clogs
If any error code appears, based on the type of error code the test case is failed.

If the error code starts with 8 it depicts Unit crash- considered as FATAL error. If error code starts with 7, it depicts call drops or packet drops have occurred- considered as SEVERE error. If error code starts with 3 only packet drops have occurred-considered as minor error.

The output of Clog analysis is as shown in Fig9.

UNIT NAME	CLOG INFORMATION	VERDICT
DMPG-0		PASS
DMPG-1		PASS
DMPG-2		PASS
DMPG-3		PASS

Fig 9: Analysis output of Clogs

If any exception occurs then, the information column would contain the description of exception and verdict would be fail.

3. Alarms analysis:

The alarms logs collected are as shown in Fig 10.

```
RNC       IPA2800                    2015-02-05  15:31:58

ALARM HISTORY


NBT (26107) IPA2800          1AG01-00-16     PROCES       2015-02-05 15:00:05.41
    DISTUR  CMD-0
    1222    DATA WRITING VIA VD8 DEVICE FAILED
    SLTPRB  CMD-0
    01004 01 135346

NBT (26108) IPA2800          1AG01-00-16     PROCES       2015-02-05 15:00:07.42
    DISTUR  CMD-0
    1222    DATA WRITING VIA VD8 DEVICE FAILED
    SLTPRB  CMD-0
    01031 01 135346
```

Fig 10: Alarm Logs

In the analysis phase active alarms which have occurred during and after the call are checked. Only if specified alarms have occurred during the call, for Eg**:**"Running out of memory buffer"," unit restarted" etc the test case is failed. The analysis output in a successful scenario is as shown in fig 11.

ALARM INFORMATION	VERDICT
	PASS

Fig 11: Analysis of Alarm Logs

The results of all the log analysis will be consolidated and put into a Testcase_name.ODS file, which has different sheets for each analysis.

Once test case level analysis is done, then all test case results are consolidated in another ODS file at test group level (Collection of test cases) and it is shown as in Fig 12.

SL. NO.	TEST_CASE_NAME	TIME	VERDICT	MEMSTAT	CLOG	ALARM
1	HSDPA16K_175_ULservicespetHSDCH_DSP	2015-02-20-01-09-05	FAIL	FAIL	PASS	PASS
2	HSDPA128K_97_ULservicespetHSDCH_DSP	2015-02-20-02-10-54	FAIL	FAIL	PASS	PASS
3	Max_Number_of_PS_NRT_64_64	2015-02-20-03-11-49	FAIL	FAIL	PASS	PASS

Fig 12: Consolidated Analysis at Test group level

IV. OUTCOME/CONCLUSION and FUTURE SCOPE

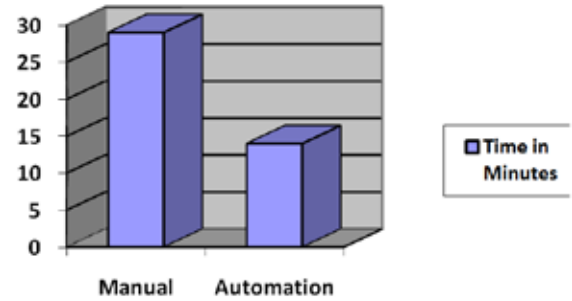

Table 1: Comparison of test case run time between Manual and Automation testing

It can be concluded from Table1 that considerable reduction of 15 to 20 min per test case in testing time is achieved by Automation Testing as compared to Manual Testing.

Further it can also be seen that since the Analysis of logs are done locally, there is no waiting time between 2 test cases i.e. the next test case execution can be started immediately after the previous test case log collection step is over.

Hence the Aim of the Paper to reduce the run time and also to overcome the manual errors is successfully achieved.

Future Scope: To develop improvised version of the tool to perform Online Analysis of the Logs. By performing this step, Run time can be further reduced and Analysis results can be known during the call run.

V. Acknowledgment

I would like to express my humble regards to Nokia Networks for allowing me to carry out project work and also R.V.College of Engineering by providing the needed help and guidance to complete the work.

REFERENCES

[1]Gokce Gorbil, Omer H. Abdelrahman, "Modeling and Analysis of RRC-Based Signalling Storms in 3G Networks", IEEE transactions on emerging topics in computing, No 99, pg 1-14, 07 January 2015. | [2] C.M.H. Noblet*, R. Ruedat, J.M. Garciat "Enabling UMTS end-end performance analysis", Fifth IEEE International Conference on 3G Mobile Communication Technologies, PG 29-33, 2004. | [3] Yingying Chen†, Nick Duffield, Patrick Haffner "Understanding the Complexity of 3G UMTS Network Performance", IFIP Networking Conference, pg 1-5, 22-24 May 2013. | [4] A. Krendzel, Y. Koucheryavy, J. Harju, "Method for Estimating Parameters of 3G Data Traffic", Vol 7, IEEE Communications Society, pg 4312-4316, 20-24 June 2004 | [5] ChangYou Xing, Guomin Zhang, Ming Chen " Research on Universal Network Performance Testing Model" International Symposium on Communications and Information Technologies (ISCIT 2007),Pg 780-784,July 2008. | [6] Muhammad Dhiauddin Mohamed Suffiani, Fairul, "Performance Testing: Analyzing Differences of Response Time between Performance Testing Tools", ICCIS, IEEE, Pg919-923, 2012.