ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 6, Issue 5, May 2017

DEVELOPMENT OF VALIDATION FRAMEWORK FOR ABIS INTERFACE IN GSM BTS

C.M.Kousar Nazneen, Dr.S.Ravishankar

Abstract**—** The need for reducing the Operational Expenditure for the telecom customers has led to the increased capacity of GSM carriers per BTS hardware. To satiate the same requirement for the existing customer's lots of software optimization was done to increase the GSM carrier capacity. This paper discuss about how to develop a framework for testing the complete functionalities of the Abis Interface. Once the framework is developed new test case will be created and manually tested. Goal of Automation is to reduce number of test cases to be run manually and not eliminate manual testing all together.In order to extract the packet Information the tools required are Wireshark and the Splitcap parser.It creates the awareness of different types of the development, compilation, code repository tools, testing and their execution flow. By using this framework it is easy to identify the issues the failure test cases and the unexpected scenarios within a less time. The result of this paper include codecoverage report which shows the efficiency of the framework.

IndexTerms**—**Abis,Code Coverage ,Splitcap**.**Wireshark

I. INTRODUCTION

In prior days, the meaning of media transmission is telephone and telegraph, that allowed the people to communicate at a distance by voice , and telephone service was provided by the public switched telephone network (PSTN)[1]. PSTN uses analog technology, which varies from one nation to other and from one manufacturer to next manufacturer.

In the mean time, the requirement for telecommunication services were remarkably increased.so the Global System for Mobile communications (GSM)came in to existence. First GSM specified by 900 MHz range[1]. Later the GSM adjusted the 1800 and 1900 MHZ band of frequencies[3]. The most used technology for voice and data communication which is having more than 2 billion subscribers everywhere through out the world is GSM. For the packet oriented mobile data service on the communication system General Packet Radio Service(GPRS) is used[1].

The Abis interface lies within base station subsystem (BSS) and it represents the dividing line between the BSC(Base station controller) and the BTS. The BSC and BTS are connected using leased lines, radio links or metropolitan area network (MANs). BTS is a component which performs channel encoding , encryption/decryption and is comprised of radio transmitters and receivers, antennas etc.

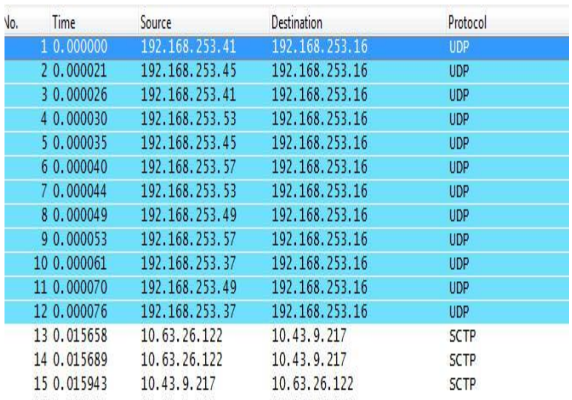
As the technology is developing day by day , GSM technology allows for one BTS to host up to 16 TRXs. It is challenging for any company to satisfy customers requirements and to deliver quality products with less expenditure. Once the Product is released in the market the company should maintain the proper quality of service.

So the Framework is produced to test the stand alone scenarios without implementing hardware. It diminishes the minimum usage of code and reusability of the code.It reduces the dependency on hardware. Debugging is easy. When a test fails, only the latest changes need to be debugged. With testing at higher levels, changes made over the span of several days, weeks or months need to be scanned.

The paper is organized as follows. In Section 2, tools used for extracting the packest is explained. In Section 3 developing of unit testing framework is described. In section 4 Codecoverage results of the work are shown. Finally, the paper is concluded in Section 5, and future scope is discussed in Section 6.

II. TOOLS USED FOR EXTRACTING THE PACKETS

Firstly to develop the framework of ABIS,circuit switch logs and packet switch logs are collected by using live wireshark Troubleshooting,optimization,security can be done by using the wireshark tool[2]. It is able to configure a TCP/IP network. Wireshark have logging tools ,logfiles can be captured weekly or hourly rate based on the requirement of network and capability of handling devices as shown in fig. 2.1.



No.	Time	Source	Destination	Protocol
1	0.000000	192.168.253.41	192.168.253.16	UDP
2	0.000021	192.168.253.45	192.168.253.16	UDP
3	0.000026	192.168.253.41	192.168.253.16	UDP
4	0.000030	192.168.253.53	192.168.253.16	UDP
5	0.000035	192.168.253.45	192.168.253.16	UDP
6	0.000040	192.168.253.57	192.168.253.16	UDP
7	0.000044	192.168.253.53	192.168.253.16	UDP
8	0.000049	192.168.253.49	192.168.253.16	UDP
9	0.000053	192.168.253.57	192.168.253.16	UDP
10	0.000061	192.168.253.37	192.168.253.16	UDP
11	0.000070	192.168.253.49	192.168.253.16	UDP
12	0.000076	192.168.253.37	192.168.253.16	UDP
13	0.015658	10.63.26.122	10.43.9.217	SCTP
14	0.015689	10.63.26.122	10.43.9.217	SCTP
15	0.015943	10.43.9.217	10.63.26.122	SCTP

Fig. 2.1 Wireshark Capture of packets

To separate the packet information split cap parser is used.Based on the IP addresses ,specific port number

ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 6, Issue 5, May 2017

packets are extracted according to our requirement. It splits one big pcap file into multiple files based on TCP and UDP sessions, one pcap file per session.Commands for reading the pcap files from wireshark logs is as shown in fig.2.2.

```
OPTIONS:
-r <input_file> : Set the pcap file to read from.
                  Use "-r -" to read from stdin
-o <output_directory> : Manually specify output directory
-d : Delete previous output data
-p <nr_parallel_sessions> : Set the number of parallel sessions to keep in
   memory (default = 10000). More sessions might be needed to split pcap
   files from busy links such as an Internet backbone link, this will however
   require more memory
-b <file_buffer_bytes> : Set the number of bytes to buffer for each
   session/output file (default = 10000). Larger buffers will speed up the
   process due to fewer disk write operations, but will occupy more memory.
-s <GROUP> : Split traffic and group packets to pcap files based on <GROUP>
   Possible values for <GROUP> are:
             flow        : Flow, i.e. unidirectional traffic for each 5-tuple,
                           is grouped together
             host        : Traffic grouped to one file per host. Most packets
                           will end up in two files.
             hostpair    : Traffic grouped based on host-pairs communicating
             nosplit     : Do not split traffic. Only create ONE output pcap.
   (default) session     : Packets for each session (bi-directional flow) are
                           grouped
             seconds <s> : Split on time, new file after <s> seconds.
             packets <c> : Split on packet count, new file after <c> packets.
-ip <IP address to filter on>
-port <port number to filter on>
-y <FILETYPE> : Output file type for extracted data. Possible values
   for <FILETYPE> are:
             L7   : Only store application layer data
   (default) pcap : Store complete pcap frames
-z : Lazy file creation, i.e. only split if needed
-recursive : Search pcap files in sub-directories recursively
```

Fig. 2.2 Commands of the Split Cap

III. UNITTEST FRAMEWORK

The text file can be extracted by writing script in any language according to our convenience. The test file obtained is as shown in the fig.3.1.

```
0a08000001013d00a004140000a0702116530a2bcf8f097b885c4edfdd8102
0a08000001013d00a004140000a0702116530a2bcf8f097b885c4edfdd8102
0a08000001013d00a0041401014070212caa5dc8d5ba55a277799bd165a94e
0a08000001013d00a0041401014070212caa5dc8d5ba55a277799bd165a94e
0a08000001013d00a0041401014070212caa5dc8d5ba55a277799bd165a94e
0a08000001013d00a004140201e070210c0c7e150dae4e0f8bfc8db4cd7045
0a08000001013d00a004140201e070210c0c7e150dae4e0f8bfc8db4cd7045
0a08000001013d00a004140201e070210c0c7e150dae4e0f8bfc8db4cd7045
0a08000001013d00a0041403028070212887 4fc562045edffca0f92cb644b9
0a08000001013d00a00414030280702128874fc562045edffca0f92cb644b9
0a08000001013d00a00414030280702128874fc562045edffca0f92cb644b9
0a08000001013d00a00414040320702133 9a8e112b7185a23687c87bbfd8bc
0a08000001013d00a004140403207021339a8e112b7185a23687c87bbfd8bc
0a08000001013d00a004140403207021339a8e112b7185a23687c87bbfd8bc
0a08000001013d00a004140503c07021336d73241ab01bb0880bad2e2d9ad8
0a08000001013d00a004140503c07021336d73241ab01bb0880bad2e2d9ad8
0a08000001013d00a004140503c07021336d73241ab01bb0880bad2e2d9ad8
0a08000001013d00a0041406046070210a5653fe88c8607be0337dc0b09d06
0a08000001013d00a0041406046070210a5653fe88c8607be0337dc0b09d06
0a08000001013d00a0041406046070210a5653fe88c8607be0337dc0b09d06
0a08000001013d00a00414070500702135800f9d49955aa9ac38520f9295c5
0a08000001013d00a00414070500702135800f9d49955aa9ac38520f9295c5
0a08000001013d00a00414070500702135800f9d49955aa9ac38520f9295c5
0a08000001013d00a004140805a07021107fd791f3cf5bbe3d7d9f2625ce25
0a08000001013d00a004140805a07021107fd791f3cf5bbe3d7d9f2625ce25
0a08000001013d00a004140805a07021107fd791f3cf5bbe3d7d9f2625ce25
0a08000001013d00a0041409064070211f1eb60ae4fafc65cac132197ad82e
0a08000001013d00a0041409064070211f1eb60ae4fafc65cac132197ad82e
0a08000001013d00a0041409064070211f1eb60ae4fafc65cac132197ad82e
0a08000001013d00a004140a06e070213bcf0cb909f905f6adffcee749a64c
0a08000001013d00a004140a06e070213bcf0cb909f905f6adffcee749a64c
0a08000001013d00a004140a06e070213bcf0cb909f905f6adffcee749a64c
0a08000001013d00a004140b0780702111ea05aa8a748ce231e13c5e2b5861
0a08000001013d00a004140b0780702111ea05aa8a748ce231e13c5e2b5861
0a08000001013d00a004140b0780702111ea05aa8a748ce231e13c5e2b5861
0a08000001013d00a004140c0820702121929a81112340644a5bb1db24d253
0a08000001013d00a004140c0820702121929a81112340644a5bb1db24d253
0a08000001013d00a004140c0820702121929a81112340644a5bb1db24d253
0a08000001013d00a004140d08c070213d373ade6e6b314ed99062d9b608c5
```

Fig.3.1Test file of the packets given as input to the framework.

Test cases are run for different type of codec modes such as full rate speech and half rate speech[3].The testfile contaims the information about MUX header,RTP header and payload bits.

For the development of Abis Unit Test Framework[4], initially we need to compile the Abis as an independent module in Linux platform without any dependency from the other modules.

Each line in the text file is considered as individual packet and given as input to the Abis framework. The characters in the line are converted to bytes and perform the downlink operations . The message that we get in the Downlink would be looped back to the Abis module by calling the Uplink function .

Once the uplink operations are completed the message is identified based on the message Id of the message derived from the message Id field. This message is stored in a .CSV file output

I V. CODECOVERAGE

To determine the percentage of code that was executed during the test process, code coverage is required. GCOV is used for codecoverage in linux operating system[5].The code coverage completely depends on the make file. make is a linux tool which is used for building the program in a simplified way in which the objects or executables were built. In the single make file source flags ,linking flags are declared.To measure the effectiveness of testing efforts gcov is used.To understand the analysis of code coverage report is required.gcovr is used to analyse the programs compiled with GCC..It generates a simple html output as shown in fig.4

`GCOV` uses two files for profiling. The names of these files are derived from the original object file by substituting the file suffix with either `.gnu`, or `.gcda`. The .gcno file is generated when the source file is compiled with the GCC -ftest-coverage option. It contains information to reconstruct the basic block graphs and assign source line numbers to blocks.

The .gcda count data file is generated when a program containing object files built with the GCC -fprofile-arcs option is executed.A separate .gcda file is created for each object file compiled with this option. It contains arc transition counts, value profile counts, and some summary information.

Compilation of main.c can be done by using this commands:

$(ARC) $(OBJS) $(CFLAGS) $(MOD_FLAGS) $(LDFLAGS) -o $(EXEFILE)

$(CC) $(CFLAGS) $(MOD_FLAGS) -c -fprofile-arcs -ftest-coverage $<

Commands to generate gcov html report

mkdir $(OUT_DIR)

gcovr -r $(BASE_DIR) . --html -o

$(OUT_DIR)/coverage.html --html-details

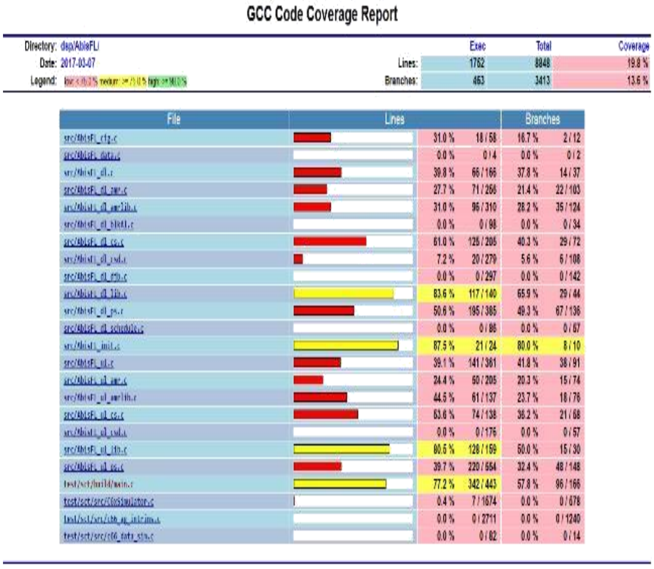


Fig. 4.1 CodeCoverage report

ACKNOWLEDGEMENT

I sincerely thank Electronics and Communication Engineering Department of R.V. College of Engineering, Bangalore for providing the opportunity and guidance for research work.

REFERENCES

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".
[2] Wolf-Bastian Pottner, L.W. IEEE 802.15.4 packet analysis with Wireshark and off-the-shelf hardware.
[3] 3GPP. Full Rate Speech Transcoding. Series 06: GSM Codecs Specification 06.10, European Telecommunications Standards Institute, Dec 1999.
[4] ITU-T recommendation G. 1000 (2001), Communication quality of service: A framework and definition
[5] Code Coverage Analysis http: //www.bullseye.com/coverage.html

C.M.Kousar Nazneen

received her B.Tech degree in Electronics & Communication Engineering from KSRM COLLEGE OF ENGINEERING (JNTU, Anantapur) kadapa, A,P., India in 2014 and is shortly finishing her M.Tech degree in Communication Systems from R.V. College of Engineering (VTU,Belgaum), Bangalore, Karnataka, India. Her interest areas are advanced wireless communication, digital signal processing.

Prof. Dr. S Ravi Shankar

entered the field of education in the year 2004 after 32 years of exemplary service in the R&D industry , He earned his Bachelor's degree in Engineering (Electronics) from BITS PILANI and Masters in Technology from IIT karaghpur in the year 1985. He also has a ph.D from IIT madras in Microwave Communication & Radar .He took premature retirement from the Qualcomm in the year 2004 and entered the field of education. He is the professor. (Electronics and Communication) in RV Engineering College in Bangalore. Affiliated to VTU, Belgaum. His interest areas are Electro Magnetics, Wireline & wireless broadband, Underwater Communication.