Abstract

LTE (Long Term Evolution) is a 4G wireless communications standard created by the third Generation Partnership Project (3GPP). It is intended to give up to ten times the speeds of 3G systems for cell phones, for example, smartphones, tablets, netbooks, and remote hotspots. LTE is developed from an existing 3GPP framework known as the Universal Mobile Telecommunication System (UMTS). In principle, LTE can be up to ten times speedier than 3G. In practice, the actual system speed will change when different components, for example, system load, and signal quality are taken into account. LTE is made out of numerous new advances compared with existing generations of mobile frameworks.

LTE-Advanced fundamentally upgrades the previous LTE Release and supports much higher peak rates, higher throughput and scope, and lower latencies, bringing about a better client experience. In existing system, TP simulator establishes RRC connection and executes various scenarios for single or multiple RNTIs (users). There are some drawbacks in the existing system like we have to run/create initial cell setup before running any test case, deleting UE context manually, etc. Also, we have to integrate the code manually and after each change, we have to test all the scenarios manually which consumes more time and this integration process takes few months duration. The objective of this project is to develop Test Environment for E-UTRAN RRC (Radio Resource Control) layer and to enhance existing TP simulator/stub.

The E-UTRAN consists of Wipro eNodeB (Base Station eNB), providing the E-UTRA user plane (PDCP/RLC/MACPHY) and control plane (RRC) protocol terminations towards the UE. The goal of this project is to reduce the manual activities and to automate the entire process of building and unit/integration testing using continuous integration (CI) with Jenkins.

Keywords: 3G, 4G, LTE, UMTS, Mobile, Continuous integration, Jenkins.
1. Introduction

The ITU-R has set guidelines for 4G connectivity to a set of standards which defines connectivity and speed. For mobile use, including cell phones and tablets, we need connection speeds with minimum 100 Mbps and for steady uses like LAN or hotspots, minimum 1 Gbps. At the point of declaring these standards, the speeds were an objective for engineers, a point that denoted a significant jump over the present technology. The minimum speeds discussed for 4G were a bit unreachable in the beginning regardless of the measure of money put into the project by the manufacturers to accomplish the target. Accordingly, the directing body chose that the name of the technology used should be LTE [Long Term Evolution] and could be named as 4G if it gave a significant change over the 3G technology. As for now, LTE is the fastest technology available in the market for wireless communications.

This new technology has grabbed the attention of everyone as it provides much improved performance which is up to 50 times better than the existing technologies. It has better efficiency than the cellular networks. In real-time application like Voice over IP, HD video streaming, conferences or faster connections, LTE is very useful. LTE supports MIMO [Multiple Input, Multiple Output] transmissions which will allow the base stations to transfer data streams simultaneously in the same carrier. LTE is composed of various components. The main components are:

- **User Equipment [UE]**: A device which is used for any communications by an end-user. It establishes the connection with the base station. The interface for this connection is known as LTE-Uu.

- **Evolved UMTS Terrestrial Radio Access Network [E-UTRAN]**: It takes care of radio communications between EPC and cellular device. The base stations are responsible for controlling cellular devices in different cells.

- **Evolved Packet Core [EPC]**: It starts communication with PDN [Packet Data Networks] over the internet or in private corporates or IP interactive media subsystem.

Every base station or eNodeB makes connection with packet core with the help of an interface known as S1AP [S1 Application Protocol]. It can also establish a connection with the neighboring base stations with the help of an interface called as X2AP [X2 Application Protocol]. This protocol is used for forwarding the packets during the handover phase. Radio Resource Control [RRC] is an interface which helps to establish connection between eNodeB and UE. All the messages are exchanged and all the communications happen over RRC interface.

1.1. LTE Packet Life

The flow or a lifecycle of a packet is started from physical layer and then it enters MAC layer and so on. MAC layer executes transport as logical mapping. In short, it is a function that separates distinctive logical channels in transport
block for the other layers. The information required by RRC for dealing with the whole network is provided by measurements and format selection. Above MAC layer, there are some logical channels which serve as services for data transfer. Below the MAC layer, there are channels in transport blocks called as transport channels. These channels are used to represent services for transferring data. The different logical channels used here are:

- **DTCH**: Dedicated Traffic Channel is an end-to-end channel which is also exclusive for each UE. It exists in downlink as well as in uplink. It is used to transfer user information.
- **BCCH**: Broadcast Control Channel is used only in downlink for transferring system Control Information to all.
- **PCCH**: Paging Control Channel is used to transfer paging information in downlink. It is used when network is unaware of location of UE.
- **CCCH**: Common Control Channel is an uplink channel which is used to exchange control information between network and user.
- **DCCH**: Dedicated Control Channel is a bidirectional channel which is used to transfer information from network to UE and vice-versa. It can only be used by users who have RRC connection.

The RLC layer is used to perform segmentation and reassembly. The RLC guarantees in order delivery of SDUs. The expired packets can be sent during handover. PDCP [Packet Data Convergence Protocol] layer only for circuit switched bearers and also for packet data bearers which have established the connections directly between the host and RLC layer. The user-plane PDCP functions contain decryption, Sequence numbering, removing the duplicates and header decompression.

### 1.2. Working of LTE

LTE has unique call flow which is defined by 3gpp standards. LTE call flow is nothing but creation of signals and various sessions which happens between different components of LTE namely **UE** [User Equipment], **eNB** [evolved base station], **MME** [Mobile Management Entity], **HSS** [Home Subscriber Server], **SGW** [Serving Gateway] and **PGW** [PDN Gateway]. **LTE** is the fastest service available in the current scenario. It uses more radio spectrum in each connection and achieves the high speeds. In order to build an LTE, many elements are needed like antennas in the Radio base station known as eNodeB\(^{10}\). The **MME** is used to establish connection with UE and to control the signalling with the terminal. All the data is sent and received in the form of IP packets using LTE. The data is taken from user equipment and sorted out in different packets to an eNodeB radio base station. Transport network sends the data from eNodeB to the gateway which has many levels. The SGW routes and sends the data on further level. The
network gateway [PGW] looks at the destination of these packets and then it has to reach its final destination though
the internet and other IP networks.

2. Literature Survey

The various layers\cite{1} in the LTE protocol stack are:

- **Physical Layer:** This layer conveys all the information from MAC transport channels. i.e. the air interface.\cite{5} It also takes care of finding the cell for the handover purpose and also for synchronizations.

- **MAC [Medium Access Control] Layer:** This layer maps the logical channel to transport channels. MAC layer takes care of multiplexing and de-multiplexing of SDUs from various channels.

- **RLC [Radio Link Control] Layer:** RLC Layer is responsible for transfer of upper layer PDUs, error correction with the help of ARQ, segmentation as well as reassembly of RLC SDUs.\cite{1} RLC is also responsible for re-segmentation of RLC data PDUs, also, ordering of RLC data PDUs is done again, and duplicates are detected, RLC SDU discard, RLC re-establishment, and protocol error detection.

- **RRC (Radio Resource Control) Layer:** The fundamental functions of RRC sub layer consists of broadcasting the system information [SI] which speaks about the NAS [non-access stratum] and AS [access stratum].

- **PDCP [Packet Data Convergence Protocol] Layer:** This Layer handles the compression as well as decompression of various headers such as IP data. It eliminates the duplicate SDUs for layers beneath PDCP.\cite{4}

- **NAS [Non-Access Stratum] protocols:** This is the highest stratum between user or UE and MME in the control plane. NAS supports the portability of UE and also session management strategies to set up and maintain connectivity of an IP between the user equipment and PDN gateway.

In the existing system, each scenario and working of each layer should be checked manually and the sequence of call flow should be verified in each scenario. This approach leads to wastage of time and resources. This integration process includes many steps and it takes around several months or sometimes a year to complete the whole process.\cite{8}

So, as a result, we can say that integration is time consuming and an unpredictable process. Integrating a project or the work of all the developer’s is a complex process as problems can arise while merging the code and it may not the intended output for the project. To overcome this problem, programmers came up with an idea of integrating the code daily but it is not possible manually. Manually integrating the code daily would take at least a week and that is not a feasible solution. So, in turn a tool can be developed and used for this process. There are many tools which were developed to overcome this problem of integration process.\cite{6}
2.1. Practices of Continuous Integration

- Maintaining single source code repository for all the developers
- Automating the build process
- Developing code for making build as self-testing
- Committing in the repository everyday
- Building all commits on the integration machine
- Any failed builds should be fixed rapidly

2.2. Tools used for CI

- **Buildbot** - This tool is written in python language. Buildbot is dependent on twisted system. Python setup script is necessary for the master to handle the Buildbot configuration. This might be somewhat hard for newcomers who are not into development, yet such scripts are useful for flexibility.

- **Travis CI** - There are some drawbacks such as this tool hangs and stops the build for no apparent reason and it clones Git repository even when there are no changes made in GitHub. So, the build starts over and over again even when nothing is happening. Many times, we get broken build just because of some bug in CI framework. Even if a single test fails and the rest are successful, it will declare the build as broken and it won’t give any warnings or messages.

- **Bamboo** - This is a continuous integration [CI] server and supports builds in various programming languages. Build notices or messages can be altered based on type of event and can be sent using various post build options like e-mail or SMS. This tool also supports only limited SCM repositories.

- **BuildMaster** - This is an automated tool which helps in the deployment process. It merges components to handle the automation process basically related to CI. BuildMaster is also helpful in production deployments. Its list of features and scope of use allows it to work with the DevOps development, and is showcased as a simple way for programming shops to keep up with the DevOps procedure.

- **Jenkins CI** - Jenkins is an open source CI platform which is written in Java. This project was derived from Hudson after a debate with Oracle. Jenkins gives CI services for developing different software. It is based on server framework running in a servlet holder, for example, Apache Tomcat. It supports almost all the available SCMs in the market such as CVS, GitHub, Subversion, Mercurial, Git, Perforce, RTC, AccuRev and Clearcase.
Jenkins can also execute Apache maven and Ant based tasks. It can also execute shell scripts and windows batch commands.

Jenkins is created because of the dispute that happened between the contributors of Hudson and the company that owned Sun microsystems. As we know, this tool is developed in Java. So, it can also be installed using command line after downloading WAR file from Jenkins website and by giving a simple command ‘java -jar jenkins.war’. We can also send it to servlet containers. The main focus of Jenkins is to build and test software projects persistently and notice the jobs which are running externally on the server. As we have seen previously, this tool supports various SCM repositories like Subversion, CVS, GitHub, Mercurial and many more. There are different plugins available which can do almost everything that is needed for the project. All the configurations can be set up according to the project needs. Once the build is done, the results can be sent to the developers via e-mail.

3. Proposed System

3.1. Transport Plane Enhancement:

In this paper, we are focusing on the enhancement of product so that it will be able to deliver the speeds as mentioned in theory. In existing system, Transport Plane simulator sends responses to the requests received from Control Plane [CP]. The entire configuration, start requests are sent from CP and Transport Plane responds to each request accordingly. But all this process can be done only for one scenario at a time. For testing different test case, we need to kill the previous process manually and start the next test case manually. Also, before running each test case, we need to perform the initial cell setup and then run the next test. The context created in the previous test should be deleted manually in order to execute next test case successfully.

In this project, we are going to eliminate all these drawbacks so that all the tests will be run without any manual interruption and the results will be stored on the server. At times, a message in a call flow is not sent or received and other message is sent twice. But while verifying, the Transport Plane simulator does not count it as a duplicate message. So, as a result, the test case is shown as successful when the actual result should be failure. To fix this bug, we are going to modify the existing program.

3.2. Continuous Integration using Jenkins:

In the next module, which is continuous integration, we are going to install and configure Jenkins in order to automate the whole integration process as well as building and unit/integration testing is also automated. Previously, this integration process used to take few months to complete the whole process i.e. to merge the source code by all
the developers and keep it in one place called as code repository. Once the code is merged, we have to carry out build and tests manually to verify if there are any conflicts in the code. If there are any conflicts, then we need to find out which code change is causing that conflict and to do so manually, it would take longer duration which is not recommended. So, to save the efforts, we need to come up with a solution which will minimize the manual interaction and which will automate the build as well as testing environment in Transport Plane [TP]. The best way to implement this is to find a continuous integration tool which is best suited for this project. There are many tools in the market but many has some or the other limitations. One of the best Continuous Integration tool available is Jenkins. In this project, we are going to use Jenkins and customize it according to the project needs in order to automate the entire process of building, running the unit tests and sending the results to the developers.

To implement this, we need to write scripts which when triggered, can automate the entire process. Also, we need to modify the code to fix the bugs and to execute it with Jenkins. In this project, we are going to automate the setup for UE [or RNTI tests] and MME [Mobile Management Entity] related tests. The purpose of this project is to enhance the existing product and to automate the project as much as possible.

4. System Design -

The UE needs to start the Attach Process. Once the Attach is successful, a context is built up for UE in the MME. Once the context is established, a default bearer is set up from the UE to the PDN gateway and an IP address is assigned to it.

- At first, the connection between UE and eNodeB is established by UE. Then, the UE transmits Attach Request message along with PDN Connectivity Request on the RRC connection which is established initially. This request is sent for PDN or IP connectivity. After this, S1 logical connection with MME is established by the UE.
- The identity of UE is provided in the attach request message but if UE is not identified by the network then the Authentication and Security mode procedures are initiated.
The location of UE is given to HSS by MME with the help of Update Location Request message. Once this is done, the database is updated by HSS with the present location of the UE. It transmits the subscriber profile information to MME. Then, MME sends create session request [eGTP-C protocol] to the serving gateway.

A default bearer is created for this UE by the serving gateway. The S-GW sends the request to PDN-GW to make a bearer for UE between the serving gateway and PDN gateway in order to provide end to end connectivity.

On receiving this request, PDN gateway creates the bearer and provides an IP address to the UE. Then, serving gateway sends a create session response to the MME.

A bearer is needed between eNodeB and serving gateway which will be established by the MME. MME sends the ‘S1AP initial context setup request’ to eNodeB in order to create the UE context.

When the initial context setup request is received, eNodeB starts the AS security mode command process to establish the security parameters with the UE.

Once this is done, the security parameters are established by the UE. These parameters are needed for ciphering integrity protection. The UE sends security mode complete message to eNodeB.

The resources available with UE need to be reconfigured by eNodeB. This can be done by sending RRC connection reconfiguration request to UE. Then, the UE responds back with RRC connection reconfiguration complete message and before sending this message, it upgrades the RRC connection configuration.

The eGTP-C modify bearer request is sent to the serving gateway by MME. It is sent so that the eNodeB tunnel ID will be updated by the default bearer.

Once the information is upgraded, serving gateway replies with modify bearer response to MME. After this, the MME sends attach accept as well as activate default bearer context request NAS message to UE.

If GUTI is allocated by MME, at the time of sending attach accept, it should be processed by the UE and respond back with Attach complete.
1. The first step in continuous integration is taking the code from the latest branch in the code repository. The copy of code in the local developer’s machine is known as working copy. Once, the working copy is available, developer can modify the code as per the requirement or developer can add new files to the code. Once the modifications or addition is done, the developer should build and test the code in the local machine. If the build and tests are successful that means, the code changes are done successfully. After doing this, the code can be pushed into the repository which is common for all the developers.

2. In Jenkins configuration, there is an option for polling the SCM in particular intervals. Once the changes are committed, Jenkins will Poll the repository and it will indicate the changes.

3. These changes will trigger the build automatically. This build will check out the recent changes in the local developer’s machine. It will also start the build on the local machine or server and will execute the test cases.

4. Once the build is completed, Jenkins will report the results and the console output will be shown in the Continuous Integration server.

5. The build can be successful or failure depending on the changes made in the code. There can be conflicts in the code of two developers. Jenkins keeps the log of each commit along with the date, time and name of the developer who has made changes in the code repository.

6. These reports are collected and the feedback is sent via e-mail or message to the Team Leads or Managers or Clients. Once the report is sent, further action is taken by the Managers.

5. **Results and Discussion -**

![Successful outcome of a Test-case.](image)

The above snapshot is the successful test case. If the test fails, we can collect the logs, check for the failed call flow and correct the call flow. If all the expected messages are received in the correct order, only then the test will be successful.
In the above image, Master-Slave configuration is done in Jenkins. This is helpful when any developer commits the changes in the repository while the build is going on. Now, if there is only one instance, the developer will have to wait for the previous build to get over. With the help of multiple slaves, master can trigger the build and post it on any idle machine. The builds can run parallel for the same project or for different projects. This can be done by adding the nodes from Manage Jenkins > Manage nodes > New Node and setting the configuration. In the Configuration section, check the ‘Execute concurrent builds if necessary’ box. This will setup the master slave configuration to run two or more jobs in parallel depending on the resources.

LTE can be divided in two planes namely Transport Plane [TP] and Control Plane [CP]. In this project, Transport Plane simulator is enhanced and some new scenarios are written to test the call flows from TP to Control Plane and vice-versa. Also, the bugs from the existing system are removed and manual handling is minimized. This project also includes Continuous Integration using Jenkins and complete automation of the system. The results obtained in this project are as follows:

- In the existing system, many scenarios were failing due to some bugs and wrong Call Flows. Those bugs were found and fixed so that the previously failed test cases will run successfully.
- TP Stub code is modified in such a way that it will display all the test cases and user can select any test case to run.
- A function named ConfigXmlParserRnti() is written in order to avoid running initial call setup every time before any test case is executed. With the help of this function, initial cell setup will be done automatically and the UE related test cases can be run continuously without any interruption.
- A small thread [MsgBitmap()] is added in order to avoid duplicate messages. If any duplicates are there, the test case will fail else if all the messages are received, it will be successful.
Timer is added in the TP stub so that whenever a test case is idle for certain amount of time, it will exit the loop and declare the test case as failed.

Jenkins is installed on various servers for Continuous Integration setup. Existing TP stub code was modified according to Jenkins usage.

Different scripts are written to automate the entire process of creating branches, checking out the code, building the binaries and running the test cases.

Jenkins is configured and Post Build steps were added to send build results to the team leads and managers and each test case result is displayed separately.

Existing MME stub code is modified so that it will work with the modified TP stub and new test case is added which will combine the Setup, Modification and Release call flows for ERAB setup with MMESTUB.

Scripts are written to automate the build process for TP stub and MME stub and to run the tests for UE as well as MME. Integration tests are also run along with these test cases.

DEVOPS stage I and stage II are completely automated by modifying the existing code, writing scripts and using Continuous Integration with Jenkins. Also, Jenkins is configured so that it will run two or more jobs in parallel using Master-Slave setup.

As a result, an enhanced Transport Plane simulator is developed and MME stub is also modified so that it will work with the modified TP stub. Jenkins is installed and configured to automate the whole project.

6. Conclusion

In this project, Transport Plane simulator for LTE [Long Term Evolution] is enhanced. The drawbacks in the existing system which includes 3G, 3.5G and 4G are eliminated in 4G LTE technology. In the enhanced Transport Plane simulator, manual interaction is nearly eliminated and the code is written in such a way that it will eliminate all the flaws like receiving duplicate messages, wrong sequence of messages in the call flow due to which test cases were not successful.

In this project, TP simulator is designed to run all the test cases continuously without any interruption.

This project also automates the whole process of development right from checking out the code, committing the changes in the repository, building the project and running the unit as well as integration tests using Continuous Integration with Jenkins. Thus, the entire development process and working of Transport Plane simulator is automated in this project.
7. Future Work

This project includes enhanced Transport Plane simulator. The future scope of this project includes enhancement of Control Plane by finding the root causes of the bugs and fixing the bugs. The data rates and speed will be much higher than the existing system in the later release of the LTE. The automation of development process is done in this project. On a later stage, the entire production process will be automated by writing more scripts and using Jenkins automation. The master slave configuration is done using few resources but later, this configuration will be done in more than hundreds of machines so that developer will be able to build and test the scenarios as soon as changes are committed. The system will be efficient so that all the scenarios will be executed successfully within few minutes and there will be complete utilization of time and resources.

References

1. “Evolved universal terrestrial radio access (E-UTRA); Physical channels and modulation (release 10)”, TSG RAN
2. “Evolved universal terrestrial radio access (E-UTRA); Radio resource control(RRC); Protocol specification (release 10)”, TSG RAN
3. “LTE-Advanced RRC specifications for different protocols”, www.sharetechnote.com