ENSC 894: COMMUNICATION NETWORKS

Simulation of General Packet Radio Service (GPRS) Network

Final Project – Spring 2014

Sathappan Kathiresan

Email address, skathire@sfu.ca

April 20, 2014

Abstract

In this report, we will upgrade an existing OPNET GPRS model to OPNET v 16.0.A. The model captures the signaling and transmission behavior of the GPRS network. The enhancements to the existing model are the implementations of the Logical Link Control (LLC) layer, the Base Station Subsystem (BSS), and the cell update procedure. We first present an overview of the GPRS network, the LLC layer, and the existing OPNET model. We first discuss about the different GPRS models developed, we then address the difference between these models and their implementations in OPNET. The debugging facilities that OPNET Modeler provides are of two levels- object level debugging and Source level debugging. Source level debugger like CDB (Microsoft console Debugger), MSVC (Microsoft Visual C++ Debugger) and GDB (GNU Project Debugger). We will use these debuggers to resolve the conflict involved in dependencies while upgrading an OPNET model from lower version to higher version. Simulation results are compared and discussed between the lower and higher version OPNET model.

Contents

List of Figures

Introduction

1.1 Overview

Traditional circuit switched mobile networks are efficient in providing voice service. Nevertheless, they are inefficient in offering packet switched service. The second generation Global System for Mobile Communications (GSM) circuit switched network provides a relatively slow data transmission rate of 9.6 kbps. The European Telecommunications Standards Institute (ETSI) has introduced a data packet switched service known as General Packet Radio Service (GPRS). GPRS provides a packet radio access for Mobile Stations (MSs) and a packet switched routing functionality for the network infrastructure [1]. An MS may be a cellular phone or a laptop connected via a cellular phone. GPRS network belongs to a 2.5 Generation network, and it is viewed as the stepping-stone to the Third Generation (3G) network.

In this report, we will upgrade an existing GPRS model to OPNET v 16.0.A. we then discuss about the various implementation in the different GPRS models and steps involved in upgrading the model and finally we will contribute a upgraded GPRS model to OPNET model library.

This report is organized as follows: section 1 gives an overview about the GSM and GPRS architectures and introduces some related studies in this area of research. Section 2 explains the various implementations involved in the different GPRS models. Section 3 presents the simulation results and graphs along with an explanation of the results. Finally, the conclusion and some suggested future work are in section 4.

1.1.1 GPRS

The system architecture of GPRS is shown in Fig. 1. In order to enable GPRS services in the existing GSM infrastructure, two new nodes are introduced: SGSN and GGSN. Several modifications are made to the existing nodes. General Packet Radio Service (GPRS) is a packet switched service based on Global System for Mobile Communications (GSM), an extensively deployed voice technology [1], [2]. GSM networks operate at 900 MHz and 1,800 MHz in Europe. In North America, they operate at 850 MHz and 1,900 MHz. GPRS is a 2.5 G cellular network. It provides affordable and fast Internet connections to service users. Billing is based on the amount of data transferred rather than on the connection time. This is achieved by allocating resources (radio channels) to users only when they need to send data. GPRS may offer data rates up to 171.2 kbps. GPRS utilizes most nodes in an existing GSM network. Two additional nodes

are introduced in the GSM network to support GPRS: Serving GPRS Support node (SGSN) and Gateway GPRS Support Node (GGSN). These two nodes constitute the core network of a GPRS sub-network and they are connected via an IP based GPRS backbone network.

In order to access the GPRS service, an MS first makes its presence known to the network by performing a GPRS Attach. In order to send and receive GPRS data, the MS activates the packet data address that it wishes to use. This operation makes the MS known to the corresponding GGSN, and interworking with external data networks can begin. User data is transferred transparently between the MS and the external data networks via encapsulation and tunneling: data packets are equipped with GPRS-specific protocol information and transferred transparently between the MS and the GGSN [4].

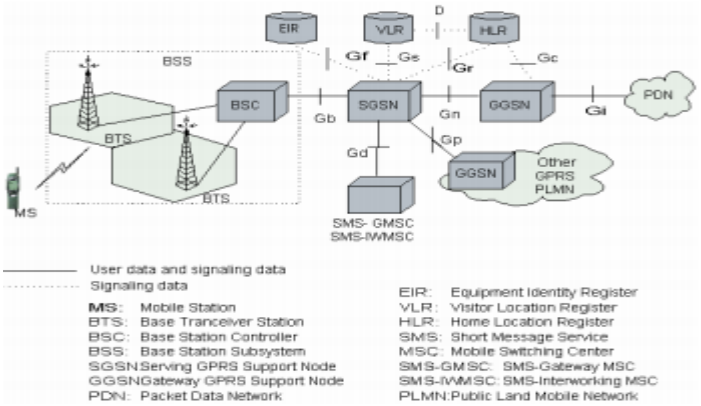


Fig 1 : GPRS Network Architecture

A GPRS network consists of the following network components shown in Figure 1: MS, BSS, Home Location Register (HLR), SGSN, and GGSN. The GPRS network is connected to an IP based network to provide GPRS users with access to packet switched data service. To access the GPRS network and to start data transmission, the MS has to execute Attach and Activation signaling procedures. User data transmission between MS and an external packet

1.1.2 Logical Link Layer

LLC, a sub-layer of layer-2 in the ISO 7-layer reference model, conveys information between layer-3 entities in the MS and the SGSN. The GPRS transmission plane illustrated in consists of a layered protocol structure. The transmission plane provides user data transfer, along with the associated information transfer control procedures such as flow control, error detection, error correction, and error recovery. The LLC is independent of the underlying radio interface protocols. This enables the introduction of alternative GPRS radio solutions.

Two modes of operation supported by LLC are: unacknowledged peer-to-peer operation, known as Asynchronous Disconnected Mode (ADM), and acknowledged peer-to-peer operation, called Asynchronous Balanced Mode (ABM).

1.1.3.1 Internal LLC layer structure

The internal components of LLC and their functions are:

1. Logical Link Management Entity (LLME) performs parameter initialization error processing and connection flow control invocation.
2. Logical Link Entities (LLEs) control the information flow of individual connections. They provide unacknowledged (ADM) and acknowledged (ABM) information transfer, flow control in ABM operation, and frame error detection.
3. When a frame is transmitted, Multiplex Procedure generates and inserts a Frame Check Sequence (FCS), performs frame ciphering, and provides contention resolution between LLEs. On frame reception, the multiplex procedure performs the frame decipher function and distributes the frame to the appropriate logical link entity after checking the FCS.

1.1.3 LLC Service Primitives

Service primitives consist of commands and respective responses associated with the services requested from another layer. The general syntax of a primitive is:

XXX - Generic name - Type (Parameters), where XXX designates the service access point between the LLC layer and the layer providing or using the service.

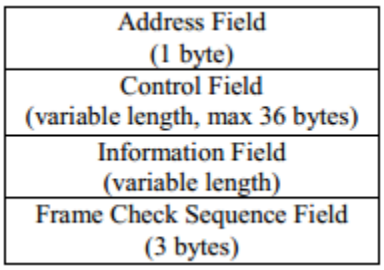
For LLC, XXX is: LLGMM for the Service Access Point (SAP) between the LLC layer and the GMM function, LL for the SAPs between the LLEs and layer 3, GRR for the SAP between the LLC layer and the Radio Link Control/ Medium Access Control (RLC/MAC) layer, BSSGP for the SAP between the LLC layer and the Base Station Subsystem GPRS Protocol (BSSGP) layer.

The four types of primitives are:

1. Request: used when a higher layer is requesting a service from the next lower layer.
2. Indication: used by a layer providing a service to notify the next higher layer of activities related to the Request primitive type of the peer.
3. Response: used by a layer to acknowledge the receipt from the next lower layer of the Indication primitive type.
4. Confirm: used by the layer providing the requested service to confirm that the activity has been completed (successfully or unsuccessfully).

1.1.4 LLC Frame Format

The format of LLC frame is shown in Figure. Each LLC frame consists of a header, trailer, and information field. The frame header consists of address and control fields and has a minimum length of 2 bytes and a maximum of 37 bytes [6].



1.2 Project Motivation

The standards LTE, 4G/3G are all evolved from GPRS and GSM. To study these standards, we will need a working model to present the end to end delay, QoS Profiles and Signaling process time. So we decided to make the model active in OPNET v 16.0.A and paved way for the model to be improved (downlink operation)

1.3 Related work.

There are several models explained the process of GPRS in OPNET.

The following are the models which I considered for my Project,

1. General Packet Radio Service OPNET Model v 11.0.A
2. Enhanced General Packet Radio Service OPNET Model v 10.0.A
3. Simulation of General Packet Radio Service OPNET Model v 9.0.A (with MAP protocol)
4. Simulation of General Packet Radio Service OPNET Model v 9.0.A

We will now discuss about each model in respect to their implementations,

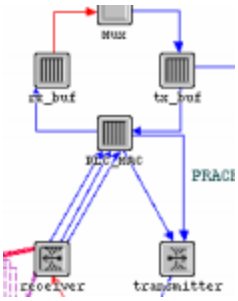
1.3.1 General Packet Radio Service OPNET Model v 11.0.A

In this model they discuss about the implementation of the Radio Link Control/Medium Access Control (RLC/MAC) and the Base Station Subsystem GPRS protocol (BSSGP). The RLC/MAC and BSSGP protocols are added to an existing GPRS OPNET model. They have enhanced the existing model [1] by implementing unacknowledged mode of RLC and two phase access mechanisms. The implementation of BSSGP enables the exchange of radio-related and data messages from Base Station Subsystem (BSS) to Serving GPRS Support Node (SGSN). They

have verified the effect of the new implementation on the end to-end delay and cell update mechanism by performing OPNET simulations. The enhanced model was tested using a network with 17 mobile stations.

1.3.1.1 Radio Link Control/Medium Access control

They have implemented the unacknowledged mode of RLC and fixed allocation medium access mode. Two phase access procedure and CS-1 coding scheme are also implemented. In the MS node model, shown in Fig. 7, the first six channels in the receiver are dedicated to receive the PBCCH information from the BTS. The uplink frequency corresponding to the PBCCH frequency is considered as the PRACH frequency. The MSs have a dedicated channel for sending packet channel requests. The Power monitor node receives the PBCCH information from the BTSs and measures the power of the received messages. It then selects the BTS with the highest power level as the serving BTS.



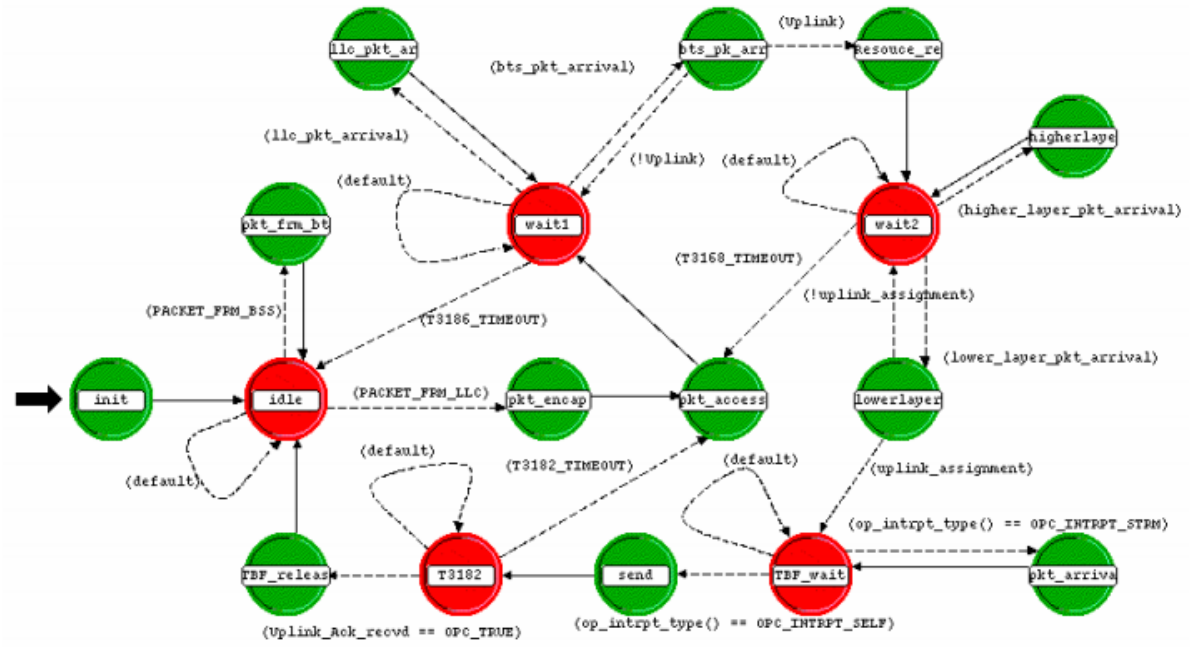
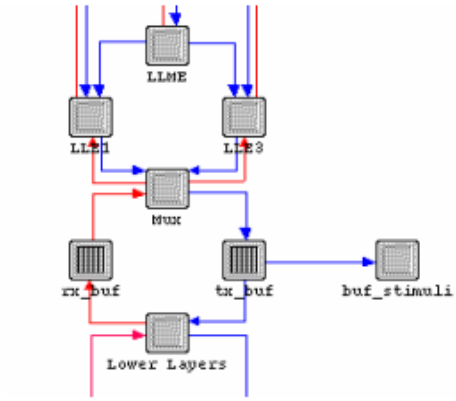


Fig 2: RCL/MAC node and Process Model

1.3.2 Enhanced General Packet Radio Service OPNET Model v 10.0.A

These enhancements to the existing model are the implementations of the Logical Link Control (LLC) layer, the Base Station Subsystem (BSS), and the cell update procedure. They first present an overview of the GPRS network, the LLC layer, and the existing OPNET model. We then describe the implementation of the LLC layer, the BSS consisting of a Base Station (BS) and a Base Station Controller (BSC), and the autonomous cell reselection procedure performed by the mobile station. Four simulation scenarios were used to verify the accuracy of the OPNET implementation



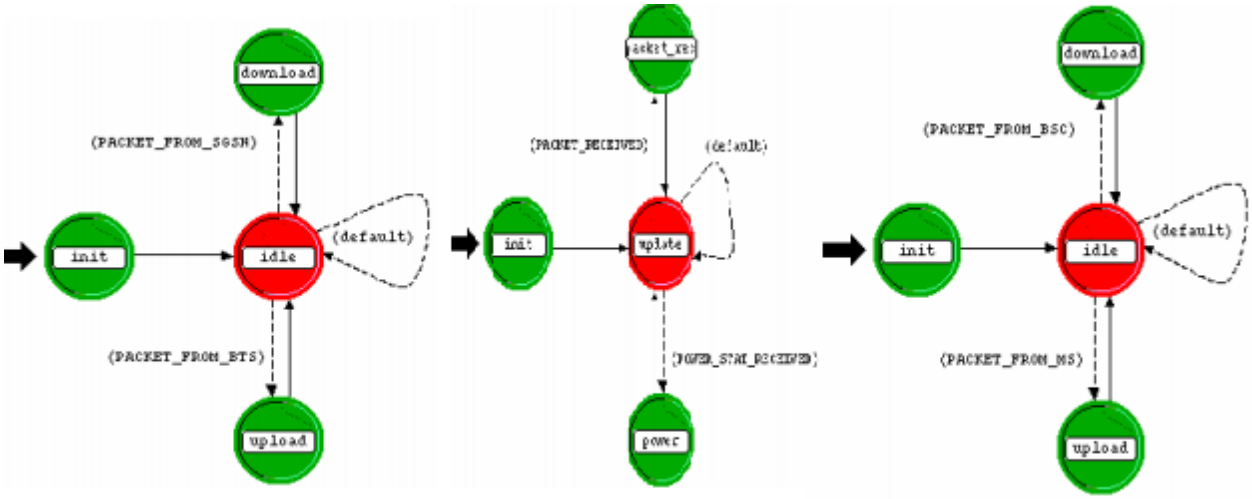
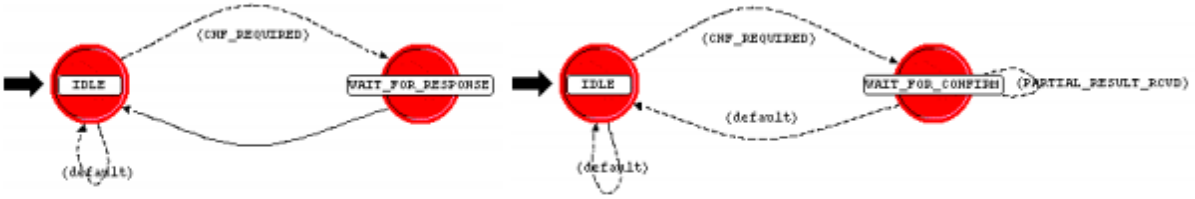


Fig 3: LLC layer and Process model of Power monitor, Bs router

1.3.3 Simulation of General Packet Radio Service OPNET Model v 9.0.A (with MAP protocol)

They have implementation of the Mobile Application Part (MAP) protocol within the General Packet Radio Service (GPRS) model. MAP represents an application layer protocol residing on top of the Signaling System 7 (SS7) protocol stack. In GPRS networks, MAP protocol supports signaling exchanges with Home Location Register (HLR) and Equipment Identity Register (EIR). They begin with a brief introduction to SS7 protocol stack and GPRS architecture and then describe MAP features related to GPRS and modifications that we implemented in the GPRS OPNET model to provide signaling capabilities for communication between HLR and the Serving GPRS Support Node (SGSN)



1.3.4 Simulation of General Packet Radio Service OPNET Model v 9.0.A

In this OPNET model of a General Packet Radio Service (GPRS) network. The model captures the signaling and transmission behavior of the GPRS network. We first introduce a GPRS network and describe the signaling and transmission procedures that will be modeled. In order to point out the simplifications made in the OPNET model, we then address the differences between the GPRS OPNET model and the standard. We also describe the implementation of each model component: node model, packet format, process model, and state variables corresponding to each component. Furthermore, we illustrate the use of OPNET to abstract the signaling and transmission behavior of the GPRS network

2 Upgrading OPNET model

2.1 OPNET programming in C++

OPNET provides a programming language called Proto-C to allow users to model various systems. Proto-C preserves generality by incorporating all the capabilities of the C/C++ programming language, i.e., a user can program in Proto-C in a similar way to in C/C++, and in Proto-C you can use the libraries that are accessible to C/C++ as well. On the other hand, Proto-C provides a set of its own APIs to model communication networks. Proto-C supports modeling

with the state transition diagram (STD) method, which makes it possible to accurately describe most systems. Figure 12.1 shows that Proto-C allows you to use both C/C++ and Proto-C libraries and write models in both C and C++ styles. In Figure 12.1, Proto-C APIs, C functions, and C++ methods are mixed together. Figure 12.2 shows the state transition diagram that models the simplest on/off system. System state transits to on or off depending on the trigger conditions. The Proto-C code can be embedded in each state. If a state is triggered, the Proto-C code within that state may be executed in response. By combining Proto-C and state transition diagrams, complex systems can be modeled.

```
strm_index = op_intrpt_strm ();
pktptr = op_pk_get (strm_index);
switch(strm_index)
{
  case 0:
  {
    // Proto-C
    op_pk_send(pktptr, 0);
    // C
    sprintf(str, "%d", ++sent_count);
    // C++
    w.write(std::string("sending␣-␣") + str);
  }
  break;

  case 1:
  {
    // Proto-C
    op_pk_destroy(pktptr);
    // C
    sprintf(str, "%d", ++received_count);
    // C++
    w2.write(std::string("receiving␣-␣") + str);
  }
  break;
}
```

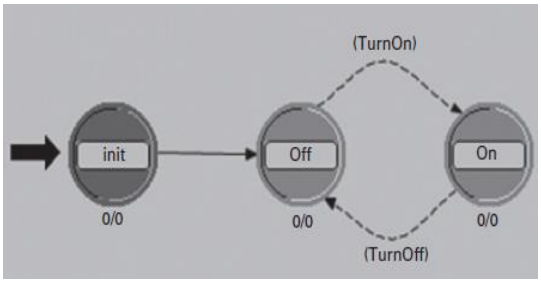


Fig 4: Process model

```
OPC_COMPILE_CPP

#include <iostream>
#include <string>
#include <vector>
#include <sstream>
#include <boost/tokenizer.hpp>

#define STREAM (op_intrpt_type () == OPC_INTRPT_STRM)
```

2.2 Debugging Simulation.

OPNET Modeler provides two levels of debugging capability: object-level debugging and source-level debugging. In object-level debugging, *object* refers to an OPNET simulation entity like packet, event, process, etc. The object-level debugging process follows the order in which discrete events are scheduled; therefore, it is normally used when you want to track the simulation on an event-by-event basis and track and inspect the simulation objects associated with these events. Object-level debugging reflects the internal logic of discrete event simulation. In source-level debugging, *source* refers to the simulation source code. The source-level debugging process follows the execution of source code; therefore, it is suitable if you want to track and watch the value of variables and inspect the details of your code. The object-level debugging technique is specific for debugging event-based simulation programs, and the source level debugging technique is for debugging general software programs. In practice, it is advisable to combine both debugging techniques in order to produce more reliable models. An object-level debugger is integrated with OPNET Modeler itself. It is called OPNET Simulation Debugger (ODB). For a source-level debugger, most general source code debuggers can be used to debug OPNET programs, such as Microsoft Console Debugger (CDB), Microsoft Visual C++ Debugger (MSVC), and GNU Project Debugger (GDB). CDB and MSVC are source-level debuggers on Windows platforms and GDB is generally used on Linux platforms .Among them, CDB and GDB can be used in conjunction with OPNET debugger, i.e., source code debugging operations can be perform with CDB and GDB within the OPNET debugging window. For

MSVC, source code debugging operations are performed within either MSVC's own Integrated Development Environment (IDE) or command line instead of the OPNET debugging window (See www.microsoft.com for Microsoft Visual C++ and Microsoft Console Debugger; www.gnu.org/software/gdb for the GNU Project Debugger).

2.2.1 Prerequisites for debugging

Before debugging simulation, certain preferences need to be set. In the "Edit" menu, choose "Preferences" to show Preferences Editor. The following preferences will be set appropriately:

1. Search for "Simulation Kernel Type" preference and set its value to "development". This is to make simulation in development mode so that the simulation program will not be optimized and debugging information, pro□ling data, and symbols will be reserved.
2. For CDB debugging, search for "Show Console Window" preference and set its value to "TRUE".
3. For CDB debugging, search for "Path to ??-bit Windows Command-line Debugger"(?? can be either 32 or 64 depending on CPU and OS support on target machine) and set its value to the path of CDB executable □le (cdb.exe).
4. Search for "Compilation Flags for Development Code" preference. For CDB and MSVC debugging, set its value to "/Zi /Od". For GDB debugging, set its value to "-g". This is to include debugging information in the compiled object file and turn off all optimizations.

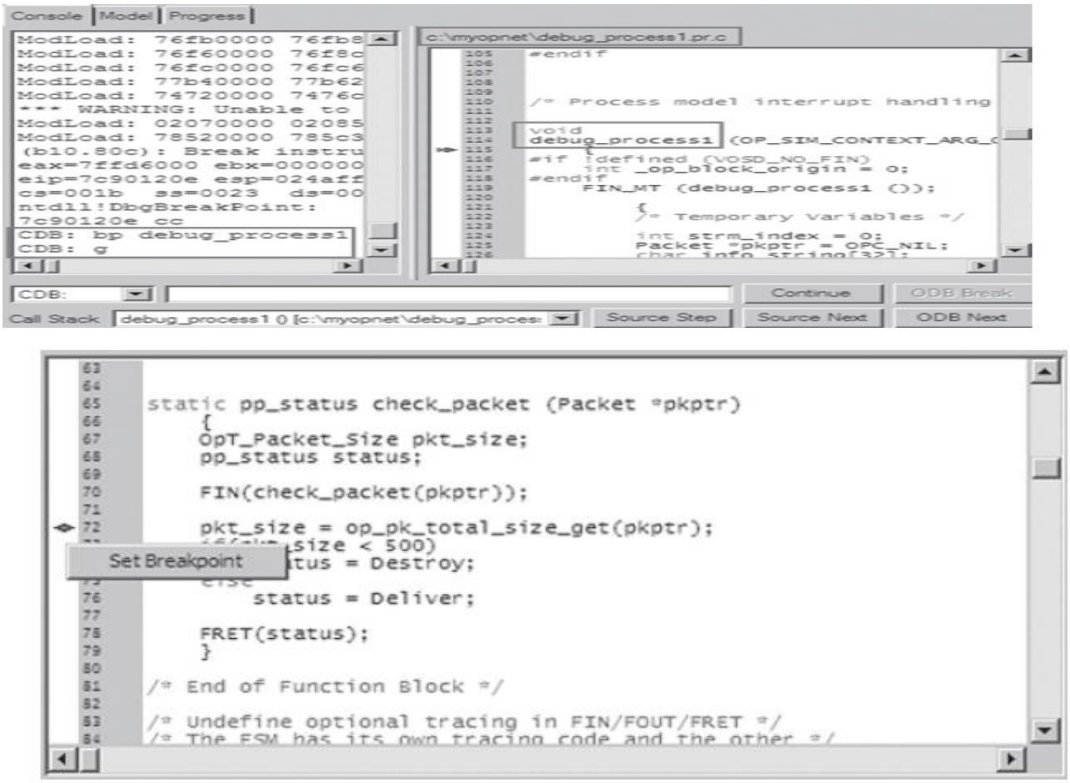


Figure : OPNET modeler Debugging window

3. Updated OPNET model

3.1 Enhanced General Packet Radio Service OPNET Model v 10.0.A

We successfully updated 24 models out of 24 models to OPNET v 16.0.A.

Debugging report is attached at the end of this report

3.2 Simulation of General Packet Radio Service OPNET Model v 10.0.A

We fully update this model to OPNET version 16.0.A. Running model and simulations are validated and results compared are same as the original OPNET model v 9.0.

3.2.1 GPRS model Project window

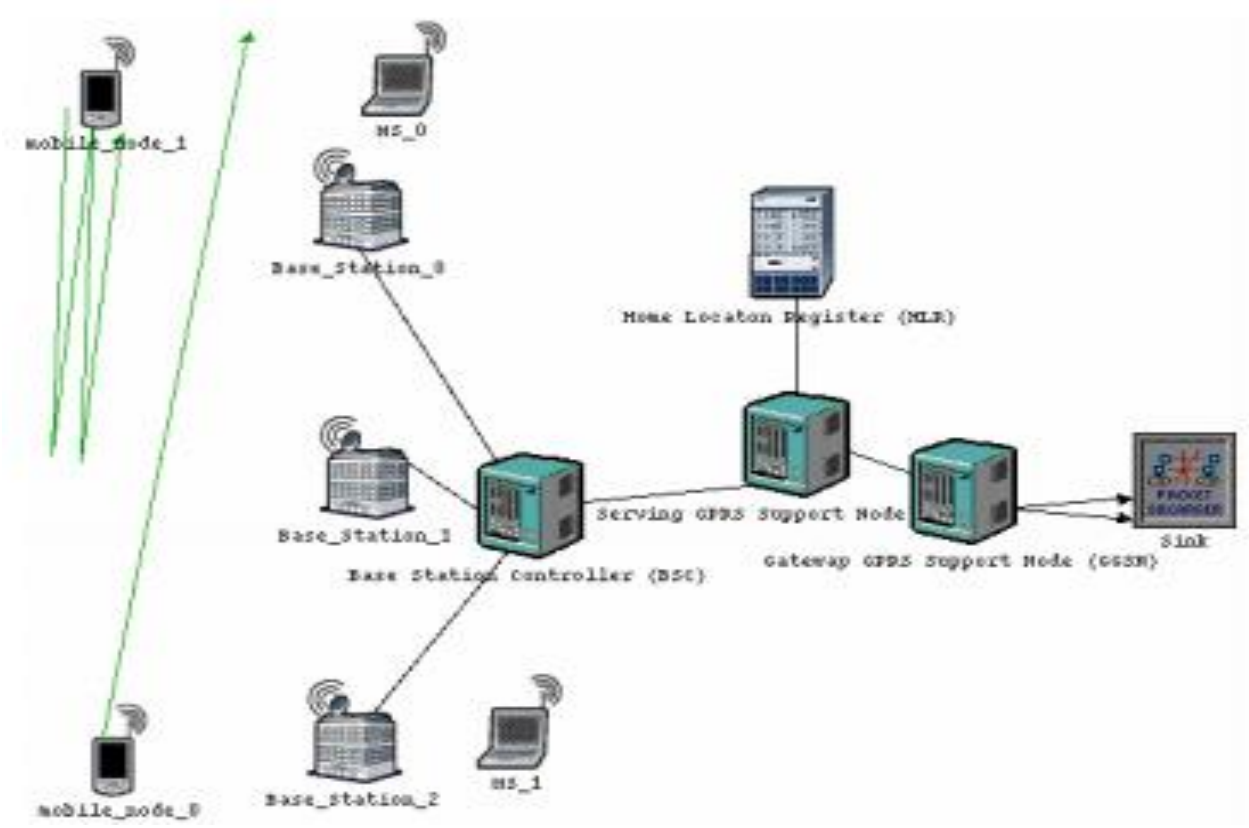


Figure : Project window of the OPNET MODEL

4 Results and Analysis

We will compare the receiver throughput between the new and old models

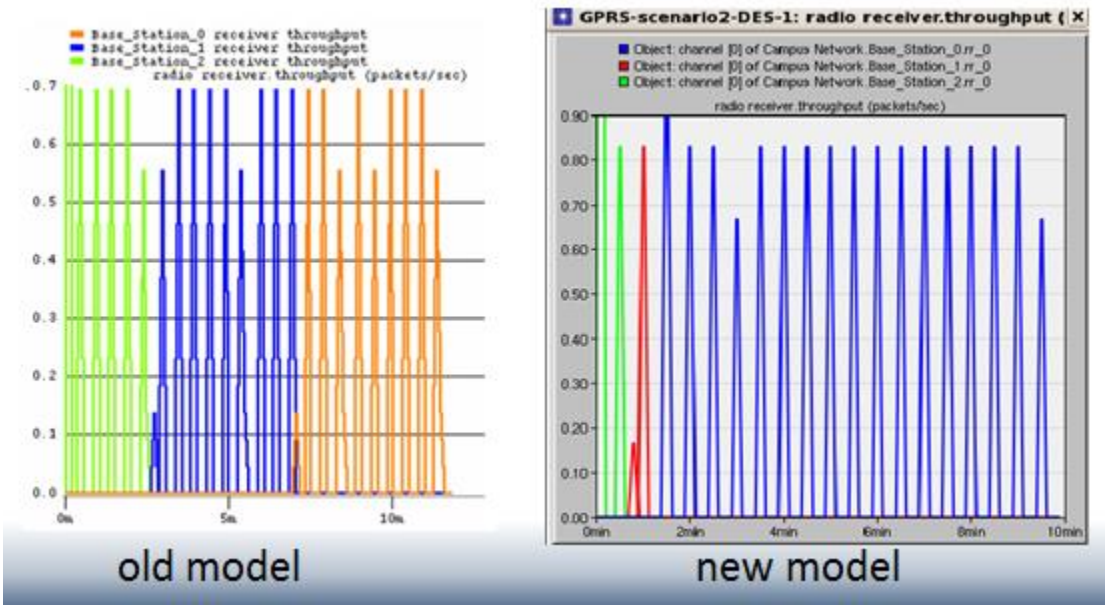


Figure: Throughput

4.1 Mobile station using Slow link and Fast link

Receiver throughput between base station 1 and 2 of mobile node_0

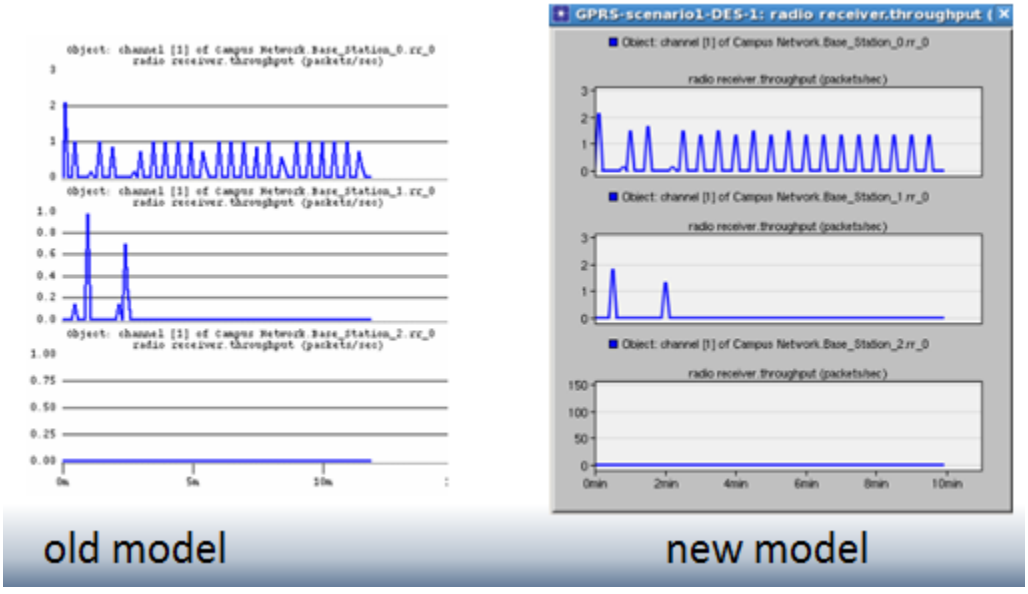


Figure: Receiver throughput between base station 1 and 2 of mobile node_0

Conclusion and Future work.

In this Project, we successfully updated a GPRS model to OPNET v 16.0.A and updated (24 of 24) to OPNET v 16.0.A. The simulation performed is compared with the simulation results of the original GPRS model.

References:

[1] Ricky Ng and Ljiljana Trajkovic, ``Simulation of General Packet Radio Service Network,''*OPNETWORK 2002*, Washington, DC, Aug. 2002

[2] V. Vukadinovic and Lj. Trajkovic, "OPNET implementation of the Mobile Application Part protocol," *OPNETWORK 2003*, Washington, DC, Aug. 2003.

[3] R. Narayanan, P. Chan, M. Johansson, F. Zimmermann, and Lj. Trajkovic, "Enhanced General Packet Radio Service OPNET model," *OPNETWORK 2004*, Washington, DC, Aug. 2004.

[4] S. Hoff, M. Meyer, and A. Schieder, "A performance evaluation of Internet access via the general packet radio service of GSM," in Proc. 48th IEEE Vehicular Technol. Conf., Ottawa, ON, May 1998, vol. 3, pp. 1760–1764.

[5] Unlocking the power of "OPNET MODELER". Zheng Lu and Hongji Yang.

Source Code Compiling Report for (24 of 24) models

```
<<< Warning >>>
  * Time:      18:35:22 Sat Apr 19 2014
  * Product:   Generic Product (32-bit)
  * Package:   Vos (Virtual Operating System) / Tfile (Typed File)
  * Function:  vos_tfile_dcell_build
  * Error:     unable to read model directory
               (check mod_dirs environment attribute)
               Dir Path: /ensc/grad1/skathire/Desktop/opnetGPRS/GPRS
|-------------------------------------------------------------------------
----|
| Reading network model.                                                  |
|-------------------------------------------------------------------------
----|
<<< Warning >>>
  * Time:      18:35:23 Sat Apr 19 2014
  * Product:   Generic Product (32-bit)
  * Package:   Vos (Virtual Operating System) / Tfile (Typed File)
  * Function:  Function Name Unavailable
  * Error:     unable to read model directory
               (check mod_dirs environment attribute)
               Dir Path: /ensc/grad1/skathire/Desktop/opnetGPRS/GPRS

|-------------------------------------------------------------------------
----|
|   ____   _____   _   _  ______  _______
|
| / __ \ | __ \ | \ | || ____||__   __| Simulation and Model Library
|
| | | | || |__) || \| || |__      | |    Copyright 1986-2010 by
|
| | | | ||  ___/ | . `||_ _|     | |    OPNET Technologies,Inc.
|
| | |_| || |     | |\ || |____    | |    as a part of OPNET Release 16.0
|
|  \____/ |_|     |_| \_||______|   |_|
|
|-------------------------------------------------------------------------
----|
|                Making  Networks  and  Applications  Perform
|
|-------------------------------------------------------------------------
----|
|   OPNET Technologies, Inc. / 7255 Woodmont Av. / Bethesda, MD 20814, USA
|
|   WEB: http://www.opnet.com / TEL: +1.240.497.3000 / FAX:
+1.240.497.3001   |
|-------------------------------------------------------------------------
----|
```

```
| Protected by U.S. Patent 6,820,042.
|
|-------------------------------------------------------------------------
----|
| Network Simulation of: GPRS-scenario1
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Simulation process ID is 23273
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Kernel: optimized, sequential, 32-bit address space
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Opening animation history file.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Reading network model.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Reading result collection file: (GPRS-scenario1)
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Verifying model consistency.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Rebuilding scenario model library.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (LLE1_process) [1 of 23]
|
```

```
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (LLE1_process) compilation successful.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (LLE3_process) [2 of 23]
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (LLE3_process) compilation successful.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (LLME_process) [3 of 23]
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (LLME_process) compilation successful.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (Mux_process) [4 of 23]
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (Mux_process) compilation successful.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (Power_monitor) [5 of 23]
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (Power_monitor) compilation successful.
|
```

```
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (abcGateWaySink) [6 of 23]
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (abcGateWaySink) compilation successful.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (bsc_router_process) [7 of 23]
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (bsc_router_process) compilation successful.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (bts_router_process) [8 of 23]
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (bts_router_process) compilation successful.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (ggsnProc) [9 of 23]
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (ggsnProc) compilation successful.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (gprs_dequeue_gen) [10 of 23]
|
```

```
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (gprs_dequeue_gen) compilation successful.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (gprs_lower_layer_process) [11 of 23]
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (gprs_lower_layer_process) compilation successful.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (gprs_lower_layer_process_SGSN) [12 of 23]
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (gprs_lower_layer_process_SGSN) compilation successful.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (hlr_map_dsm_process_model) [13 of 23]
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (hlr_map_dsm_process_model) compilation successful.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (hlr_map_performing_ssm_process_model) [14 of
23]   |
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (hlr_map_performing_ssm_process_model) compilation
successful. |
```

```
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (hlr_map_requesting_ssm_process_model) [15 of
23]   |
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (hlr_map_requesting_ssm_process_model) compilation
successful. |
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (hlr_process_model) [16 of 23]
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (hlr_process_model) compilation successful.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (mjohanss_pc_prio) [17 of 23]
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (mjohanss_pc_prio) compilation successful.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (ms_process_model) [18 of 23]
|
|-------------------------------------------------------------------------
----|
| Process model (ms_process_model) compilation successful.

-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
| Compiling process model (sgsn_map_dsm_process_model) [19 of 23]
|
|-------------------------------------------------------------------------
----|
```

```
|-------------------------------------------------------------------------
----|
| Process model (sgsn_map_dsm_process_model) compilation successful.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (sgsn_map_performing_ssm_process_model) [20 of
23]  |
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (sgsn_map_performing_ssm_process_model) compilation
successful. |
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (sgsn_map_requesting_ssm_process_model) [21 of
23]  |
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (sgsn_map_requesting_ssm_process_model) compilation
successful. |
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (sgsn_map_user_process_model) [22 of 23]
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Process model (sgsn_map_user_process_model) compilation successful.
|
|-------------------------------------------------------------------------
----|
|-------------------------------------------------------------------------
----|
| Compiling process model (sgsn_process_model) [23 of 23]
|
|-------------------------------------------------------------------------
----|
-------------------------------------------------------------------------
| Process model (sgsn_process_model) compilation successful
|
|-------------------------------------------------------------------------
|-------------------------------------------------------------------------
----|
```