Peer-to-Peer Traffic over LTE: Simulated Performance during Cell Crossover

ENSC 833: Network Protocols and Performance
Spring 2016
Project Report

Presented by :Shweta Mazumder, Brett Wiens, Katherine Manson
Project website: www.bwiens.sfu.ca

Project Member's SFU Emails:
smazumde@sfu.ca, bwiens@sfu.ca, kamanson@sfu.ca

Peer-to-Peer Traffic over LTE: Simulate Performance during Cell Cross-over

Shweta Mazumder, Brett Wiens, Katherine Manson

***Abstract*—A model was built using NS-3 to study peer-to-peer traffic over LTE. The model was used to investigate network performance for a VoIP call between two mobile users as one of the users moved between cellular regions. The scenario was such that one user (node 1) sent the signal over LTE to the nearest base station. This base station routed the data packets to the nearest base station on the other user's cellular network. The second base station broadcasted the signal to the second user (node 2). The nearest base stations for user one changed over the duration of their call. The network performance indicator that was studied for this scenario was the ratio of successful packets transmitted to packets sent.**

I. INTRODUCTION

The objective of our project was to demonstrate the performance of the Long Term Evolution (LTE) standard using an NS-3 model (version 3.24.) LTE was developed to use digital signal processing techniques and the Internet Protocol (IP) to increase the speed and capacity of wireless data networks [1].

The application that was used to demonstrate the performance was "peer-to-peer traffic." This is a term used to describe communication between two hosts at the application level of the protocol stack.

VoIP stands for voice over internet protocol and is a term that represents using an unreliable transport layer protocol such as UDP (User Datagram Protocol) over the internet protocol to establish a communication link. By using UDP, the delay is reduced because there is no reliability overhead.

Geographic cell regions are the base areas of coverage of one cell tower (referred to as an eNB which stands for "e-Node-B"). When hosts cross-over from one cell region to another, a hand-over occurs and performance may suffer. This is the scenario we explored in our project.

II. OVERVIEW OF RELATED WORK:

[1] – Ernst, Kremer and Rodrigues: This paper provided an overview of a Wifi simulation model that was created by the authors using NS-3.

[2] – Piro, Baldo and Miozzo described a custom-made NS-3 LTE model built to study SINR (Signal to Interference and Noise Ratio) versus distance of node from base station. Only one node was modeled.

[3] - Ikuno, Wrulich and Rupp built a Matlab simulator to study performance of LTE network (at the system level). The results of the report graphically show the macroscopic path loss (attenuation.)

[4] - Baldo, Requena, Miozzo and Kwan built and described the "Lena-X2" model of LTE in NS-3. An example study was provided in the paper of a single node (referred to as a UE which stands for "User Equipment") handover across several eNBs for Received Signal Received Power/Quality.

III. SCOPE:

The scope of our project was to study nodes creating peer-to-peer traffic during cell cross-over. The scenario was such that two nodes began to communicate and then moved to other cell regions. Existing simulation code from the NS-3 library

(Lena-X2) was used as a starting point to create our own NS-3 model. Using the model we created, we ran simulations and created data traces. These traces were examined for network performance.

The most important aspect of the project was writing the code for the model which defined the attributes of the cell crossover and monitoring the effect on the network performance.

IV. IMPLEMENTATION

The NS-3 module we developed to use for our experiments (p2p_lte_handover.cc, Appendix A) described an LTE network topology consisting of three base stations and two users arranged as shown in Figure 1. In those simulations including a handover, the first user crossed from the first to the third station, while the second user stayed connected to the second station at all times. Figure 2 shows the flow of the simulation program.

For this experiment, we measured the effect of the handover at a variety of network loads; therefore, the parameter swept for our simulations was the interval between packet transmissions. To simulate the effect of generic peer-to-peer network traffic, we used constant-bit-rate UDP echo requests and replies for our traffic. For each network load, the simulation was run both with and without a handover in order to compare the effects. A Perl script was used to automate the data collection (sweepInterval.pl, Appendix B). This script invoked the NS-3 simulator for each test case, then called a parsing Perl script used to extract the relevant data from the flow monitor XML output (parseFlowResults.pl, Appendix C). The parsing script extracted the total number of bytes and packets sent and received for each flow between the users.

The set of result files was collated by a formatting Perl script to generate spreadsheet data (formatIntervalSweep.pl, Appendix D). This spreadsheet data was used to generate the results plots.

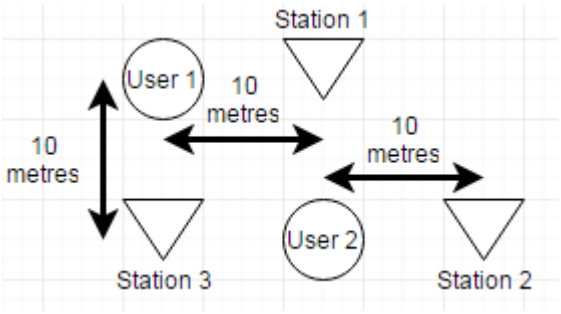

Figure 1 (Above): This figure shows the physical layout of the network topology simulated in the project. There are two users and three stations spaced according to the dimensions in the figure.

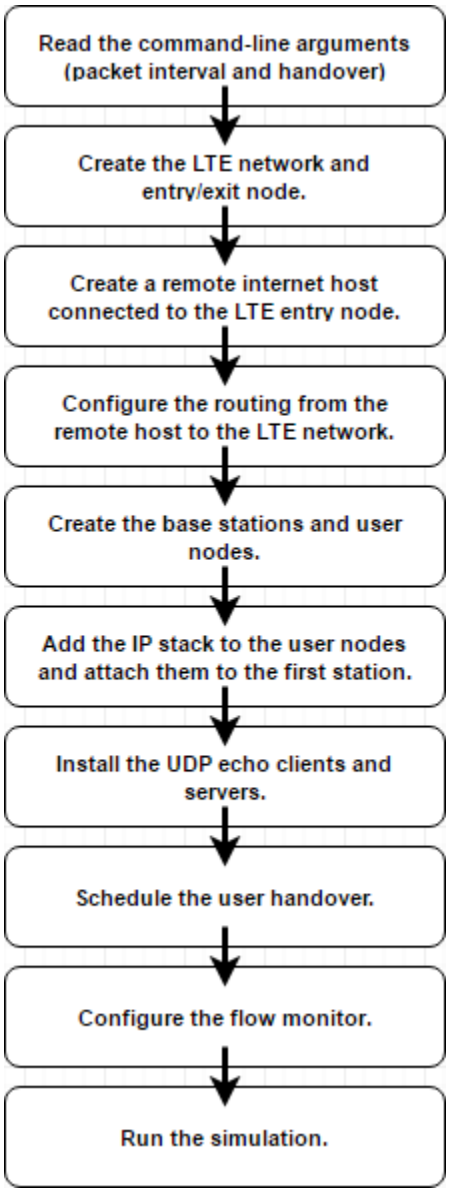

Figure 2: (Above): This figure shows the simplified model of the code created in NS-3 to simulate the handover. cases.

V. DISCUSSION AND CONCLUSIONS

The results of our simulation for UPD Echo Requests are shown in figure 3; the Packet Transmission Ratio for UDP Echo Requests. As can be seen in the graph, for this particular topology, LTE handover (the red trend line) during constant-bit-rate peer-to-peer traffic increased packet loss most of the time when compared to the no handover case (blue trend line.) This effect is mild enough that in many cases random network effects are more significant.

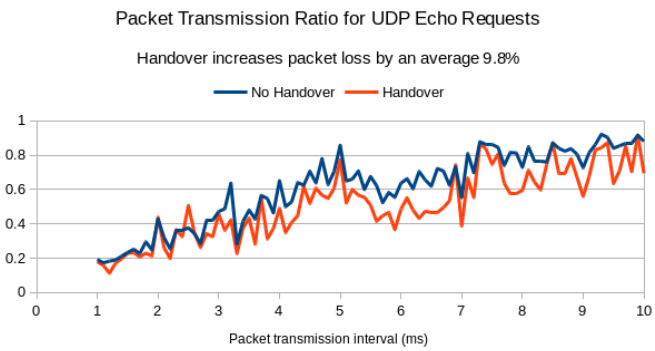

Fig. 3. Packet Transmission Ratio for UDP Echo Requests. This trend shows that as the packet interval was increased from 1 ms up to 10 ms, the ratio of successful-packets to packets-sent increased. Also the "Handover" cases in general had a lower ratio compared to the "No Handover" cases.

The server was less degraded by handover than the client as can be seen in figure 4. The Packet Transmission Ratio of UPD Echo Replies did not show a significant difference between the red and the blue trend lines. This effect is predictable; packet loss depends on traffic type; replies are significantly more likely to be successfully transmitted than request.

The overall trend of both lines in figures 3 and 4 indicate that as interval time is increased the Packet Transmission Ratio increased for both the client and the server in both handover and no-handover cases. This trend confirmed our general prediction of performance; as the time between packets increased the packet loss would decrease.

Additional packet loss was seen across all tested network traffic rates, and was not strongly correlated with the network traffic.

The main difficulty with implementing this project was in understanding the function and operation of the existing handover simulation. There were some comments in the simulation code, but for the most part, trial and error were used to determine how the code worked.

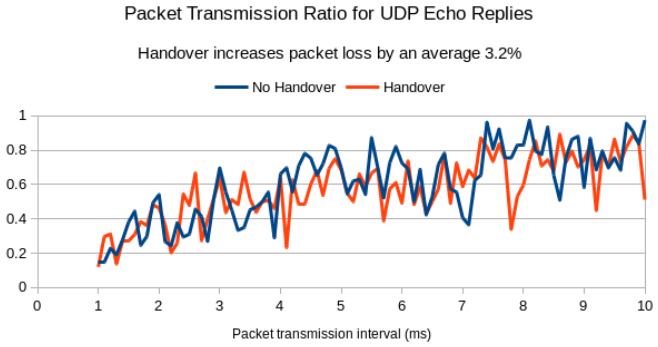

Fig. 4. Packet Transmission Ratio for UDP Echo Replies. This trend again shows that as the packet interval is increased from 1 ms up to 10 ms, the ratio of successful packets to packets sent increases. The "Handover" ratio of successful packets is closer to the "No Handover" cases.

Another difficulty with this project was avoiding simulation artifacts like simultaneous packet arrivals. This was dealt with again by trial and error.

An alternative approach to implementing our project would have been to use other network modeling software tool such as Riverbed Modeler or NS-2. The advantages of these other software tools are the animation and trace tools. We decided not to use Riverbed Modeler because it is not open source and not to use NS-2 because it is no longer being supported.

Another alternative we considered was to develop a completely different NS-3 model of a LTE handover instead of using the existing simulation model as a basis. When we started the project we felt that by using existing code we could save time and also be using proven code which would improve our results and so we decided to use the existing model. As it turns out, it is debatable that using the existing code saved time because of the time spent decoding the previous work.

Our suggestion for improvement is to further test the scalability of our model with additional UEs and eNBs. We were only able to scale our model up to two nodes.

Future work on this project could incorporate the effect of mixed internet and LTE peer-to-peer traffic on network performance.

REFERENCES

[1] Jason B. Ernst, Stefan C. Kremer, Joel J. P. C. Rodrigues, "A Wi-Fi simulation model which supports channel scanning across multiple non-overlapping channels in NS3," *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, Victoria, Canada, May 2014.

[2] G. Piro, N. Baldo. M. Miozzo, "An LTE module for the ns-3 network simulator", in Proc. of Wns3 2011 (in conjunction with SimuTOOLS 2011), March 2011, Barcelona (Spain)

[3] J.C. Ikuno, M. Wrulich, M. Rupp, "System Level Simulation of LTE Networks," Vehicular Technology Conference (VTC 2010-Spring), 2010 IEEE 71st , vol., no., pp.1-5, 16-19 May 2010

[4] N. Baldo, M. Requena, M. Miozzo, R. Kwan, "An open source model for the simulation of LTE handover scenarios and algorithms in ns-3", Proceedings of the 16th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 3-8 November 2013.

[5] NS-3 Version 3.24. [Network Simulation Software, Documentation and Lena-X2 Model]. (2016). Retrieved from www.nsnam.org.

[6] Ghassan A. Abed, M. Ismail, K. Jumari, "Modeling and Performance Evaluation of LTE Networks with Different TCP Variants," World Academy of Science, Engineering and Technology. International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering Vol:5, No:3, 2011.

[7] Md. Ebna Masum, Md, Jewel Babu, "End-to-End Delay Performance Evaluation for VoIP in the LTE network," Masters of Science Thesis. Dept of Telecom, Blekinge Inst. of Tech, Karlskrona, Sweden, June 2011.

APPENDIX A: p2p_lte_handover.cc:

```
#include "ns3/applications-module.h"
#include "ns3/config-store-module.h"
#include "ns3/core-module.h"
#include "ns3/flow-monitor.h"
#include "ns3/flow-monitor-helper.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/internet-module.h"
#include "ns3/lte-module.h"
#include "ns3/mobility-module.h"
#include "ns3/network-module.h"
#include "ns3/point-to-point-module.h"

#define N_USERS 2
#define N_STATIONS 3
// Times are in microseconds
#define T_SIM 1500000
#define T_START 1000
#define T_STOP 600000
#define T_INTERVAL 1000
#define PSIZE 576

using namespace ns3;

NS_LOG_COMPONENT_DEFINE("PeerToPeerLteHandover");

int main( int argc, char *argv[]) {
   uint16_t port_dl = 10000;
   uint16_t port_ul = 11000;
   Ptr<Node> node_lte;
   NodeContainer nodes_users;
   NodeContainer nodes_stations;
   NetDeviceContainer devices_users;
   NetDeviceContainer devices_stations;
   Ipv4InterfaceContainer interfaces_users;
   // Read the command-line arguments (packet interval and handover)..
   uint32_t interval = T_INTERVAL;
   uint32_t handover = 0;
   CommandLine cmd;
   cmd.AddValue("interval", "The interval between packet transmissions", interval);
   cmd.AddValue("handover", "Set to 1 to enable handover", handover);
   cmd.Parse (argc, argv);

   // Configure the random number generator to avoid issues
   Ptr<UniformRandomVariable> dither = CreateObject<UniformRandomVariable> ();
   dither->SetAttribute ("Min", DoubleValue (0));
   dither->SetAttribute ("Max", DoubleValue (5000));

   // Set up the LTE helper with default attributes.
   Config::SetDefault ("ns3::LteHelper::UseIdealRrc", BooleanValue (false));

   // Create the LTE network and entry/exit node.
   Ptr<LteHelper> helper_lte = CreateObject<LteHelper>();
   Ptr<PointToPointEpcHelper> helper_epc = CreateObject<PointToPointEpcHelper>();
   helper_lte->SetEpcHelper(helper_epc);
   helper_lte->SetSchedulerType("ns3::RrFfMacScheduler");
   helper_lte->SetHandoverAlgorithmType("ns3::NoOpHandoverAlgorithm");
   MobilityHelper helper_mobility;
   Ptr<ListPositionAllocator> positions = CreateObject<ListPositionAllocator> ();
   // Place the first eNodeB at (10,0,0)
   positions->Add (Vector (10, 0, 0));
   // Place the second eNodeB at (20,10,0)
   positions->Add (Vector (20,10,0));
   // Place the third eNodeB at (0,10,0)
   positions->Add (Vector (0, 10, 0));
   // Place the first user at (0,0,0)
   positions->Add (Vector (0, 0, 0));
   // Place the second user at (10,10,0)
   positions->Add (Vector (10, 10, 0));
   helper_mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
   helper_mobility.SetPositionAllocator (positions);
   node_lte = helper_epc->GetPgwNode();
   NodeContainer nodes_internet;
   nodes_internet.Create (1);
   Ptr<Node> node_internet = nodes_internet.Get (0);
   InternetStackHelper helper_ip;
   helper_ip.Install(nodes_internet);

   // Create a remote internet host connected to the LTE entry node.
   PointToPointHelper helper_pointToPoint;
```

```
   helper_pointToPoint.SetDeviceAttribute
("DataRate", DataRateValue (DataRate
("100Gb/s")));
   helper_pointToPoint.SetDeviceAttribute ("Mtu",
UintegerValue (1500));
   helper_pointToPoint.SetChannelAttribute
("Delay", TimeValue (Seconds (0.010)));
   NetDeviceContainer devices_internet =
helper_pointToPoint.Install(node_lte,
node_internet);
   Ipv4AddressHelper helper_ipv4;
   helper_ipv4.SetBase ("1.0.0.0", "255.0.0.0");
   Ipv4InterfaceContainer interfaces_internet =
helper_ipv4.Assign (devices_internet);

   // Configure the routing from the remote host to
the LTE network.
   Ipv4StaticRoutingHelper helper_routing;
   Ptr<Ipv4StaticRouting> internetStaticRouting =
helper_routing.GetStaticRouting (node_internet-
>GetObject<Ipv4> ());
   internetStaticRouting-
>AddNetworkRouteTo(Ipv4Address ("10.0.0.0"),
Ipv4Mask ("255.0.0.0"), 1);

   // Create the base stations and user nodes.
   nodes_stations.Create(N_STATIONS);
   nodes_users.Create(N_USERS);
   helper_mobility.Install(nodes_stations);
   helper_mobility.Install(nodes_users);
   devices_stations = helper_lte-
>InstallEnbDevice(nodes_stations);
   devices_users = helper_lte-
>InstallUeDevice(nodes_users);

   // Add the IP stack to the user nodes, assign IP
addresses, and attach them to the first station.
   helper_ip.Install(nodes_users);
   interfaces_users = helper_epc-
>AssignUeIpv4Address(NetDeviceContainer(devic
es_users));
   for( uint32_t i=0; i<N_USERS; i++ ) {
      Ptr<Ipv4StaticRouting> routing_staticUser =
helper_routing.GetStaticRouting(nodes_users.Get(i)
->GetObject<Ipv4> ());
      routing_staticUser->SetDefaultRoute(
         helper_epc-
>GetUeDefaultGatewayAddress(),1); }
   for( uint32_t i=0; i<N_USERS; i++ ) {
      helper_lte-
>Attach(devices_users.Get(i),devices_stations.Get(i
%N_STATIONS)); }

   // Install the UDP echo clients and servers.
   // Each user sends echo requests to each other
user.
   // Each user replies to echo requests from each
other users.
   for( uint32_t i=0; i<N_USERS; i++ ) {
      Ptr<Node> node_user0 = nodes_users.Get(i);
      for( uint32_t j=i+1; j<N_USERS; j++ ) {
         if( i==j ) continue;
         Ptr<Node> node_user1 =
nodes_users.Get(j);
         ApplicationContainer apps_client0;
         ApplicationContainer apps_client1;
         ApplicationContainer apps_server;

         UdpEchoClientHelper helper_udpDlClient(
          interfaces_users.GetAddress (i),
          port_dl+i+N_USERS*j );
         helper_udpDlClient.SetAttribute ("Interval",
TimeValue (MicroSeconds(interval)));
         helper_udpDlClient.SetAttribute
("MaxPackets", UintegerValue ((T_STOP-
T_START)/interval));
         helper_udpDlClient.SetAttribute
("PacketSize", UintegerValue (PSIZE));

apps_client0.Add(helper_udpDlClient.Install(node_
user1));

         UdpEchoServerHelper
helper_udpDlServer(port_dl+i+N_USERS*j);

apps_server.Add(helper_udpDlServer.Install(node_
user0));

         UdpEchoClientHelper helper_udpUlClient(
          interfaces_users.GetAddress(j),
          port_ul+i+N_USERS*j);
         helper_udpUlClient.SetAttribute ("Interval",
TimeValue (MicroSeconds(interval)));
         helper_udpUlClient.SetAttribute
("MaxPackets", UintegerValue ((T_STOP-
T_START)/interval));
         helper_udpUlClient.SetAttribute
("PacketSize", UintegerValue (PSIZE));
```

```
apps_client1.Add(helper_udpUlClient.Install(node_
user0));

        UdpEchoServerHelper
helper_udpUlServer(port_ul+i+N_USERS*j);

apps_server.Add(helper_udpUlServer.Install(node_
user1));

        // Start the client applications with a random
offset to mitigate simulation artifacts.
        apps_server.Start(MicroSeconds(0));

apps_client0.Start(MicroSeconds(T_START+dither
->GetValue()));

apps_client1.Start(MicroSeconds(T_START+dither
->GetValue()));
      }}

   // Schedule the user handover.
   if(handover){
     NS_LOG_UNCOND("Setting up simulation
handovers...");
     helper_lte->AddX2Interface(nodes_stations);
     helper_lte->HandoverRequest(
       MicroSeconds((T_START+T_STOP)/2),
       devices_users.Get(0),
       devices_stations.Get(0),
       devices_stations.Get(2));
   }

   // Configure the flow monitor.
   NS_LOG_UNCOND("Setting up flow
monitor...");
   Ptr<FlowMonitor> flowMonitor;
   FlowMonitorHelper flowHelper;
   flowMonitor = flowHelper.InstallAll();

   NS_LOG_UNCOND("Running simulation...");
   Simulator::Stop(MicroSeconds(T_SIM));
   Simulator::Run();

   NS_LOG_UNCOND("Exporting flow monitor
report to XML...");
   flowMonitor-
>SerializeToXmlFile("p2p_lte_handover_flowmoni
tor.xml", true, true);
   Simulator::Destroy ();
   return 0;
}
```

APPENDIX B: sweepInterval.pl

```
#!/usr/bin/perl
# Take baseline measurements without handover
for($int=1000;$int<=10000;$int+=1000) {
   print("SIMULATING NO-HANDOVER WITH
INTERVAL $int\n");
   system("./waf --run \"scratch/p2p_lte_handover --
interval=$int\"");
   system("perl parseFlowResults.pl <
p2p_lte_handover_flowmonitor.xml >
results/p2p_lte_nohandover_interval$int.out");
}
# Take experiment measurements with handover
for($int=1000;$int<=10000;$int+=1000) {
   print("SIMULATING HANDOVER WITH
INTERVAL $int\n");
   system("./waf --run \"scratch/p2p_lte_handover --
interval=$int --handover=1\"");
   system("perl parseFlowResults.pl <
p2p_lte_handover_flowmonitor.xml >
results/p2p_lte_handover_interval$int.out");
}
```

APPENDIX C: parseFlowResults.pl

```
#!/usr/bin/perl
$a=<>;
# The tags before the flow classifier contain the
results.
# Store the transmitted and received bytes and
packets by flow ID number.
while($a!~/Ipv4FlowClassifier/){

if($a=~/flowId="(\d+).*txBytes="(\d+).*rxBytes="(
\d+).*txPackets="(\d+).*rxPackets="(\d+)/){

$flowid=$1;$txb=$2;$rxb=$3;$txp=$4;$rxp=$5;
      $flow{$flowid}=[$txb,$rxb,$txp,$rxp];}
   $a=<>;};
$a=<>;
# The tags in the flow classifier contain the source
and destination information for each flow.
```

```
# Print the source and destination IP addresses,
followed by the results for that flow.
while($a!~/Ipv4FlowClassifier/){

if($a=~/flowId="(\d+).*sourceAddress="([\.\d]+).*d
estinationAddress="([\.\d]+)/){
        $addr{$1}=[$2,$3];}
   ;$a=<>;};
printf("src,\tdest,\ttxBytes,\trxBytes,\ttxPackets,\trx
Packets\n");
for($i=1;$i<=12;$i++){

printf("$addr{$i}[0],\t$addr{$i}[1],\t$flow{$i}[0],\t
$flow{$i}[1],\t$flow{$i}[2],\t$flow{$i}[3]\n");}
```

APPENDIX D: formatIntervalSweep.pl

```
#!/usr/bin/perl
for($i=1000;$i<=10000;$i+=1000){
   $nofile = `cat
results/p2p_lte_nohandover_interval$i.out`;
   $yesfile = `cat
results/p2p_lte_handover_interval$i.out`;
   @nolines = split(/\n/,$nofile);
   @yeslines = split(/\n/,$yesfile);
   print("$i\t");
   for($j=0;$j<scalar(@nolines);$j++) {
      if( $nolines[$j] =~
/^7.0.0.2,\t7.0.0.3.*\t(\d+),\t(\d+)$/ ) {
         print("$1\t$2\t"); }}
   for($j=0;$j<scalar(@yeslines);$j++) {
      if( $yeslines[$j] =~
/^7.0.0.2,\t7.0.0.3.*\t(\d+),\t(\d+)$/ ) {
         print("$1\t$2\t"); }}
   print("\n");

}
```