# CIS 549: Wireless Communications for Mobile Networks and Internet of Things

# **Project #4:** Traffic Split and Aggregation Over Multi-Radio Access Technology Simulation

Instructor: Bongho Kim (kimbong@seas.upenn.edu)

In this project you will design and implement a downlink traffic aggregation mechanism at the IP layer to utilize both the LTE and Wi-Fi networks simultaneously. This is different from a load balancing mechanism, as a single traffic flow can utilize both the LTE and Wi-Fi networks.

Please watch the project 4 demo video. It will help your understanding.

To enable the traffic aggregation **you need to build a traffic consolidation module** that performs packet in-ordering at the receiver side (which is a UE in this project) and that receives traffic from multiple networks (i.e. LTE and Wi-Fi).Also, **you need to design a traffic split algorithm** and implement it at an anchor point network element (which is the router in this project configuration.)

A close up of a map

Description automatically generated

**Figure 1. Project 4 network architecture**

Figure 1 depicts that the DL traffic flows to the LTE (blue curve) and the Wi-Fi (red curve) network starting from the router and being aggregated at the UE 4, and the UL traffic flows via LTE network only (green curve).

A close up of a map

Description automatically generated

**Figure 2. DL tunneling and IP addresses**

Figure 2, 3 and 4 are information purpose only. Figure 2 depicts the DL tunneling and IP address information configured in the provided Project #4 architecture. Each DL tunnel per UE will be in a different subnet for the sake of simplicity, so each UE IP address is assigned as following in a different subnet.

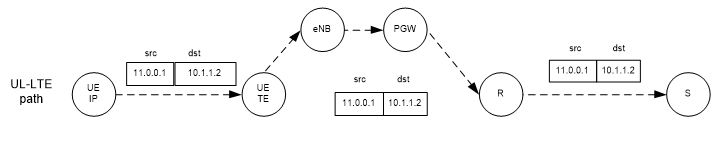
* Starting from 11.0.0.1: UE1 (11.0.0.1), UE2 (11.0.1.1), UE3 (11.0.2.1), …

The applications in each UE use the IP address in green (i.e. 11.0.0.1 in Figure 2) but when the packets to the UE from the server flows to the LTE or Wi-Fi network, the packets have to pass through a specific tunnel that is routable in the specific network. Figure 3 depicts the IP packet encapsulation for the downlink packet flow from a server to a UE, and Figure 4 shows the IP address view for the UL traffic flow.

A close up of a map

Description automatically generated

**Figure 3. DL traffic flow**



**Figure 4.UL traffic flow**

# **What to Submit:** Your implementation must be done in this **source file only:**

# **“prj4\_px.cc”** and

# The source code should be able to compile simply by copying it into the default NS-3 folder**.**

# **You also need to write one report document** in a **PDF** format for all non-coding related answers, including analysis, graphs, images and tables.

# For the submission: Create a folder name “Prj4”, and add your updated source code file “prj4\_px.cc”, and the report document. Then zip the folder before the submission. **Submit a Prj4.zip file that contains the source code file and one report**.

**Preparation**

Download a file that includes Wi-Fi and LTE architecture integrated simulation model from Coursera and copy the file in to   
your VM.

* **Step 1: copy “epc-tft-classifier.cc” and “rr-ff-mac-scheduler.cc” files to “~/ns-allinone-3.26/ns-3.26/src/lte/model” directory**
* **Step 2**: **copy “prj4\_px.cc” and “LTE\_DL\_MCS\_UE1.txt” files to “~/ns-allinone-3.26/ns-3.26/scratch” directory.   
  “prj4\_px.cc” is the main function file, and it includes supporting descriptions in multiple places.**

**You may not have to read the entire source code line by line, but read some of the comments that help you to understand the source code, at least the related areas where you will need to edit the code. These are clearly marked in the proj4\_px.cc file.**

“**prj4\_px.cc**” contains a few more input options, in addition to the options used in Project 2 and 3:

|  |
| --- |
| --Scenario=3              : Previously only two paths, "LTE path and WiFi Path" were                  provided but now traffic aggregation (LTE+WiFi) is added.                  Under the Scenario=3, there are three sub-options that                  select paths. --aggPath option is used only when Scenario=3                  is used):: --aggPath=wifiOnlyorlteOnlyorlteAndWifi  --aggPath=wifiOnly              : It uses network tunneling for the aggregation but DLtraffic uses WiFi path only.  UL traffic is configured to use LTE UL path only. --aggPath=lteOnly              : It uses network tunneling for the aggregation but DLtraffic uses LTE path only.  UL traffic is configured touseLTE UL path only. --aggPath=lteAndWifi              : It uses network tunneling for the aggregation using both LTE andWiFi path for the  DL traffic. UL traffic is configured to user LTE UL path only.  --chBwMHz=20, 10, 5, or 1.4              : Select LTE Channel Bandwidth(Default: 20). Only one value should be provided  without unit, and the unit is MHz. |

**How to use the file: “LTE\_DL\_MCS\_UE1.txt”**

**Purpose:** LTE downlink channel quality fluctuation by changing MCS value during the simulation.This feature needs to be used for validating your algorithm for the problem #2

This file contains a list of MCS value and the time duration in milliseconds (number of TTI) that will be used for the LTE Downlink traffic. This input file must be located in the /scratch directory with the main source code file, prj4\_px.cc.

The integer value (i.e. “1”) at the end of the filename (“LTE\_DL\_MCS\_UE1.txt”), before the extension, indicates the UE index. So if you created another file with the same naming convention with different index such as 2, 3, and so on, then the 2nd UE and 3rd UE in the LTE simulation will use the MCS input values from these files (if you were running multiple UE simulation). Since you don’t need to run multiple UEs for problem 1 and problem 2, you may not need LTE\_DL\_MCS\_UE2.txt or LTE\_DL\_MCS\_UE3.txt file.

* For DL MCS file for UE1: LTE\_DL\_MCS\_UE1.txt (this file is contained in the project package)
* For DL MCS file for UE2: LTE\_DL\_MCS\_UE2.txt (filename example purpose only. No file is provided)
* For DL MCS file for UE3:  LTE\_DL\_MCS\_UE3.txt (filename example purpose only. No file is provided)

**LTE MCS Input file format (1 sample file is provided for one UE)**: The first column is the MCS value and the second column is the duration in milliseconds for which the given MCS value is continuously used. Using this file, the LTE downlink MCS value will vary; MCS=28 for the first 1 sec, MCS=10 for 1.5 sec, and then go back to the first line and repeats the MCS value changes repeatedly.

You can also add more lines with different MCS values and durations.

|  |  |
| --- | --- |
| MCS Value | Duration in ms |
| 28 | 1000 |
| 10 | 1500 |

* If MCS input file does not exist in the /scratch folder, then the default MCS value (28) will be used for the entire duration of the LTE simulation.
* The first line starts from the beginning of the simulation (Frame number 0 and subframe number 0). So, each time (every 1 msec) when the scheduler is called, the corresponding MCS value from the MCS input file will be used.
* If you simply want to use MCS=28 all the time then either remove this file from /scratch directory or replace the all MCS values in the file with “28” or leave only one link with MCS=28.

**Note:**

* Don’t forget what the MCS values are in this file or remove this file if you don’t need to use to prevent an unexpected simulation output
* Remove all empty line at the end on the MCS input file.

**Example command lines for you to familiarize yourself with:**

**TCP test:  Number of UE = 1, RTT = 30 ms, WiFi\_MCS = HtMcs7, TCP RWND size = 1024000 bytes**

 Wi-Fi network test Scenario:  --Scenario=1

|  |
| --- |
| ./waf --run "scratch/prj4\_px --OutputFileName=output/prj --Scenario=1 --NumberUE=1 --Transport=1 --wifiMcs=HtMcs7 --tcpRcvBufBytes=1024000 --DataSizeforTCP=10000000 --delayValueforRHtoR=5 --delayValueforWifi=10 --simTime=7" |

LTE Network test Scenario: --Scenario=2

|  |
| --- |
| ./waf --run "scratch/prj4\_px --OutputFileName=output/prj --Scenario=2 --NumberUE=1 --Transport=1 --tcpRcvBufBytes=1024000 --DataSizeforTCP=10000000 --delayValueforRHtoR=5 --delayValueforLte=10 --chBwMHz=20 --simTime=7" |

Traffic Aggregation test Scenario:  --Scenario=3 with –aggPath= (wifiOnly, lteOnly or lteAndWifi)

|  |
| --- |
| ./waf --run "scratch/prj4\_px --OutputFileName=output/prj --Scenario=3 --NumberUE=1 --Transport=1 --wifiMcs=HtMcs7 --tcpRcvBufBytes=1024000 --DataSizeforTCP=1000000 --delayValueforRHtoR=5 --delayValueforLte=10 --delayValueforWifi=10 --chBwMHz=20 --simTime=7 --aggPath=lteAndWifi" |

**UDP test:  Number of UE = 1 UE, RTT = 30 ms, WiFi\_MCS = HtMcs7**

**For the UDP simulation, be cautious to not to use unnecessarily high “DataRateforUDP” and a long “simTime” because it may significantly slow down the simulation.**

Wi-Fi network test Scenario:  --Scenario=1

|  |
| --- |
| ./waf --run "scratch/prj4\_px --OutputFileName=output/prj --Scenario=1 --NumberUE=1 --Transport=2 --wifiMcs=HtMcs7 --DataRateforUDP=200Mb/s --delayValueforRHtoR=5 --delayValueforWifi=10 --simTime=3" |

LTE Network test Scenario: --Scenario=2

|  |
| --- |
| ./waf --run "scratch/prj4\_px --OutputFileName=output/prj --Scenario=2 --NumberUE=1 --Transport=2 --DataRateforUDP=200Mb/s --delayValueforRHtoR=5 --delayValueforLte=10 --chBwMHz=20 --simTime=3" |

Traffic Aggregation test Scenario:  --Scenario=3 with –aggPath= (wifiOnly, lteOnly or lteAndWifi)

|  |
| --- |
| ./waf --run "scratch/prj4\_px --OutputFileName=output/prj --Scenario=3 --NumberUE=1 --Transport=2 --wifiMcs=HtMcs7 --DataRateforUDP=1Mb/s --delayValueforRHtoR=5 --delayValueforLte=10 --delayValueforWifi=10 --aggPath=lteAndWifi --chBwMHz=20 --simTime=3" |

1. **Implementation and Validation [40%]:** Search for text “Problem 1” in prj4\_px.cc (2 places)

The five key items to be submitted for Problem 1 are:

1. **Flow diagram/chart** for the packet in-ordering mechanism. This must clearly show all steps how your in-ordering mechanism works or should work. Even if you could not complete the implementation, this flow diagram/chart would show your full design, so this is a very important part of this problem. Describe your algorithm clearly **using a flow diagram/chart**, but not by written description.. The “Flow diagram/Chart” should contain all your logic. Provide all possible cases that may occur in the network and show how your algorithm handles it. **If a potential problem for the packet delivery is not handled in this flow diagram/chart, it will be considered that your implementation does not handle this case.** Either a digitally designed or a hand-drawn flow diagram/chart is acceptable.

**How much detail:** Assume that you provide your flowchart to other team to implement the packet in-ordering mechanism, and consider following questions.

* Can the team implement it?
* How much the team could understand your intended mechanism?
* Is there any incorrect information in the flowchart?

1. **Implementation** of the packet in-ordering function at UE based on the flow diagram/chart. Your source code will be tested.
2. **Implement** an input option, **“inOrderTimeout”**that receives an integer time value in msec.   
   Example “--inOrderTimeout=100”. This indicates the in-order timeout value = 100ms.  The implementation should take an in-order timeout value in millisecond unit as an input, so that can be configured with different input value when a simulation is executed.. If old packet sequence number (i.e. already timed-out packet) is received, then simply pass the packet to upper protocol layer instead of dropping or delaying it. In this case, in-order mechanism does not do anything but leave it to upper layer handles it.
3. **Create** the three output files described in the next page to draw Gnuplot graphs.

**Validation** of the implementation and the functionality with various scenarios must show that the packet in-ordering function works. You must test your implementation with the various scenarios that can trigger a certain packet delivery pattern that might happen in the network and show that your in-ordering mechanism handles it as intended. **The validation is very important to understand how the algorithm behaves under certain network condition without checking the implementation.**Provide enough information to understand your validation in the report. Three example cases that you must handle are the followings. There are more cases, including multiple packet out-or-order cases, thatyou should find and handle them in your mechanism. Following list of sequences of packet numberis **just illustration purpose**, **not for you to replicate exactly**.

* + **Case 1:** Sequence number 3 arrives out-of-order and all out-of-order packets are arrived before “inOrderTimeout” expires:
    - Sequence number arrival order: 1, 2, 4, 5, 6, 7, 3, 8, 9, 10,….
    - The sequence number delivered to upper layer should be 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,…
  + **Case 2:** Sequence number 3 arrives out-of-order, but after expiring the inOrderTimeout. If inOrderTimeout=100ms, then the maximum waiting time of the packet number 4 is 100 ms. If packet number 3 does not come within 100 ms after receiving packet number 4, then packet number 4 should be delivered to upper layer. If packet number 3 arrives later then simply send it to upper layer without delay.
    - Sequence number arrival order: 1, 2, 4, 5, 6, 7, 8, 9, 3, 10,….
    - The sequence number delivered to upper layer should be 1, 2, 4, 5, 6, 7, 8, 9, 3,10,…
  + **Case 3:** Sequence number 3 does not come (lost packet): This is similar to the case above except the packet number 3 never arrives.
    - Sequence number 3 arrives out-of-order and after expiring the inOrderTimeout: 1, 2, 4, 5, 6, 7, 8, 9, 10,….
    - If, then the sequence number delivered to upper layer should be 1, 2, 4, 5, 6, 7, 8, 9,10,…

If you want to test packet loss condition, then this is one way of testing it. This is just an example and you may do your own way.

You can find similar code segment in the provided source code except the outter if-statement. The outter if-statement simply do not transmit the 100th packet. So UE will not receive the 100th packet, which means packet loss occurred.

if ( (totalPacketSent-1) != 100){

if ( totalPacketSent % 2 == 0 ) // toggle the transmission path between LTE and WiFi

{ // Use LTE path for DL traffic

m\_rtSocket->SendTo (packet, 0, InetSocketAddress (m\_msIfc0Address, TunnelPort));

}

else

{ // use Wi-Fi path DL traffic

m\_rtSocket->SendTo (packet, 0, InetSocketAddress (m\_msIfc1Address, TunnelPort));

}

}

This packet in-ordering function will be in action only when traffic aggregation scenario, “-- Scenario=3”, is used.  The example command lines for the “Traffic Aggregation test Scenario” are provided on page 3 and page 4 of this assignment sheet.

**Your validation report section must provide enough information that clearly indicates that your mechanism works** (even without testing your code).

For the validation of this problem, **use a single UE scenario** for the simplicity. If old packet sequence number (i.e. already timed-out packet) is received then simply pass the packet to upper protocol layer instead of dropping it. This information should be captured by the packet sequence number and queuing delay value in the Gnuplot output file below.

**You must create the following three output files in the “output” directory to draw Gnuplot graphsand to use them for the validation**:

* + Queuing delay (or packet waited\_time) per packet at the UE packet in-ordering receiver buffer. If a packet was directly delivered to upper layer without being delayed, then 0 mswaited\_time should be recorded.
* Format: two columns: “simulation\_timewaited\_time\_(in ms)”
* Filename: “prefix from inuput”+\_delay.dat
* Received packet sequence number at the UE before packet in-ordering.
* Format: two columns: “simulation\_timesequence\_number”
* Filename: “prefix from inuput”+\_preseq.dat
* Packet sequence number delivered to upper layer at the UE. Format (at the time of packet delivery to the upper layer)
* Format:two columns: “simulation\_time sequence-number”
* Filename: “prefix from inuput”+\_postseq.dat

1. **Implementation and Validation [40%]:** Search for text “Problem 2” in prj4\_px.cc

Implement your own packet split mechanism/algorithm to increase the TCP throughput (bps), which should be higher than the maximum average throughput achievable from the LTE or Wi-Fi network alone under the same network condition.

(Note: “throughput” unit is always “bits per sec” unless specifically indicated)

You are now replacing the temporary traffic split mechanism in the prj4\_px.cc. You need to generate DL traffic from the server to the UE, and the DL traffic split occurs at the router. A part of the packets in the traffic would flow through the LTE path and the rest of the packets in the traffic flow through the Wi-Fi path to improve the TCP throughput. The DL traffic aggregation occurs at the traffic receiver (i.e. UE).

The packet flow split module at the router must know some dynamic link performance metrics at least **link throughput (bps), one-way packet delay for both LTE and Wi-Fi path (from router to UE)**. The UE must measure this information in real-time, and this measurement should be shared with the router for a better decision making. For the sake of simplicity, you may implement the feedback mechanism in anyway you want (i.e. global variable, shared memory, etc.).

The major coding location for the implementation is marked in the source file, but you may need to update the other part of the code in other segments as needed.  Problem #2 requires using the implementations above (Problem #1). If Problem #1 is not correctly implemented, then the TCP throughput will not be enhanced.

The four key items to be submitted for Problem 2 are

1. **Flow diagram/chart** for the traffic split mechanism. This must clearly show all steps how your traffic splitting mechanism works or should work.Even if you could not complete the implementation, this flow diagram/chart would show your full design, so this is a very important part of this problem.Design your algorithm clearly **in the flow diagram/chart, not a written description**.The “Flow diagram/Chart” should contain all design specification that allows a programmer to be able to implement your algorithm. Handles all possible cases that may occur in the network and you should show how your algorithm dynamically adjust to it. Any mechanism not shown in this flow diagram/chart will be considered as not a part of your implementation. Either a digitally designed or a hand-drawn flow diagram/chart is acceptable. The flow diagram and the implementation for the algorithm must include**the real-time throughput (bps) andthe one-way packet delay (ms)measurement at the UE for both LTE and Wi-Fi path, and the traffic input bit rate measurement at the router toward LTE and Wi-Fi paths**.

**How much detail:** Assume that you provide your flowchart to other team to implement your algorithm. Then consider the following questions.

* Can the team implement it? (answer should be almost "yes")
* How much the team could understand your intended algorithm? (answer should be "clear")
* Is there any incorrect information in the flowchart?(answer should be "No")

1. **Implementation** based on the flow diagram/chart. Your source code will be tested. The implementation **must measure the following parameters and generate output files** that can be used to create the Gnuplot graphs. The output files should be in the “output” directory, and the file format for all files below is “time value\_measured”

* **Incoming link throughput at the UE per network interface; LTE (filename: prefix\_thp\_in\_lte.dat) and for Wi-Fi (filename: prefix\_thp\_in\_wifi.dat) interfaces**
* **Incoming packet delay at the UE per network interface (from the router to the UE); LTE (filename: prefix\_dly\_in\_lte.dat)and for Wi-Fi (filename: prefix\_dly\_in\_wifi.dat) interfaces**
* **Outgoing Traffic input rate at the Router toward LTE path (filename: prefix\_input\_rate\_lte.dat), and toward Wi-Fi path (filename: prefix\_input\_rate\_wifi.dat)**

1. **Validation** of the implementation and functionality with various scenarios that show the throughput improvement using both networks. The validation is to understand how the algorithm behaves under certain network condition. Without knowing the behavior of the algorithm clearly, it cannot be enhanced.
   * Your report should provide sufficient algorithm analysisfor the evaluation.**You must provide your command lines that used for the validation and the validation must use the Gnuplot diagrams or any tool that you are more familiar with using the output files that you have generated and the application throughput measurement**. You must decide the simulation scenarios (and write the run command) that can validate your algorithm and show pros and cons. You must clearly explain the intended network conditions and expected outcomes of your algorithm and compare with the actual outputs observed from the simulations.
   * To validate the dynamic behavior of your algorithm, analyze some simulationresult with the “LTE\_DL\_MCS\_UE1.txt” and your own MCS values in the file. You need to provide the MCS file and the matching simulation command line that you had used to be able to replicate the simulation results.
   * If you have tried multiple algorithms before the final one, you may also describe what the algorithms were, what you have observed and learned from it, and how you have enhanced your algorithm. This will help to understand the evolution path of your algorithm and what has been considered, which may not be visible from your final algorithm alone. This will help grading.
   * The **performance of your algorithm is expected to be**
     + “Average TCP Throughput using --aggPath=lteAndWifi” >=Max(Average TCP throughput using the “--aggPath=wifiOnly” with simulation input --Scenario=3, Average TCP throughput using “--aggPath=lteOnly” with simulation input --Scenario=3); the greater the better.

**Important:** The throughput aggregation algorithm will be **evaluated with TCP protocol and expect** various static network conditions including different combinations of LTE channel bandwidth, WiFi MCS values anddifferent network delay, and dynamic network conditions such as LTE MCS changes and packet delay in addition to the .Your link throughput aggregation algorithm **may use the “LTE channel bandwidth” and “Wi-Fi MCS value” from the input, but** your algorithm **must not use a static LTE MCS value or must not use the network delay values from the input parameter because these values could be dynamically updated during the simulation.**

**Note:** If you need to add a new input option for your algorithm, then this should be included in the command lines you provide.

1. **Analysis [20%]**

Execute the 8 simulations with the input parameters listed below and **analyze** the average system throughput results. This problem 3 can be done without completing problem 1 and problem 2.

There are two different traffic path configurations used in this problem.

* Four simulation scenarios are using Wi-Fi network (--scenario=1): both DL and UL traffic use the Wi-Fi network path.
* The other four simulation scenarios are using (--scenario=3, -- aggPath=wifiOnly). All DL packets are using the Wi-Fi path and all UL packetsare using the LTE path.

**Note**: Remove the LTE MCS text file (i.e. LTE\_DL\_MCS\_UE1.txt) in the “/scratch” directory.

**[Submit analysis report]**

**Compare the TCP average throughput results** with the results between the two network path configuration cases (“both DL and UL using Wi-Fi” and “DL using Wi-Fi and UL using LTE”). **If there was or was not a performance difference, then what would be the reason? You must explain the cause of the performance difference based on your analysis. You may reference the LWIP and LWA lecture notes.**

**The report should include all necessary Figures, Graphs, and Tables that support your analysis**. However, you should not simply dump all outputs in the report. The simulation output and graph have zero value by itself.

**[Simulation input parameters]**

There are 8 simulation scenarios out of these configurations, and the eight command lines are available in this file, **prj4\_problem3\_command\_list.txt,** provided in the project 4 package. Run these scenarios automatically, so that you can do other work while simulations are running. It may take about 1 to 2 hours depending on your PC performance. After the simulation, make sure that all UE has completed the TCP sessions. If any TCP session has not been completed until the configured “simTime” then increase the simTime and rerun the specific scenario. The default “simTime” in the file, **prj4\_problem3\_command\_list.txt**, is already set the simTime long enough, but you need to check it again, just in case.

* + Transport protocol: TCP only
  + One RWIN size: 1024 KBytes
  + One Download file size: 10 Mbytes
  + LTE MCS: 28 (default)
  + LTE channel bandwidth: 20 MHz (default)
  + Wi-Fi MCS: HtMcs7 only
  + Two Number of UE:  2 UEs and 15UEs
  + Two End-to-end RTT combination for (LTE path : Wi-Fi path): (30ms : 30 ms), (200ms, 200ms)
  + Two Network path:
    - Wi-Fi network path (--Scenario=1) and
    - Traffic circle (all DL traffic via Wi-Fi path and all UL traffic via LTE path). Use the input option “--Scenario=3 --aggPath=wifiOnly.” This path scenario is not using your split algorithm. When --Scenario=3 option is used, all UL traffic flows through the LTE path, and “--aggPath=wifiOnly” steers all DL traffic to the Wi-Fi path.