

Université d'Avignon
2018/2019

Rapport TP2 - Architecture des
réseaux



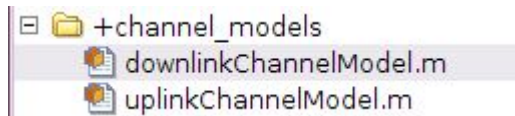
Étudiants :
Bouzid Fares

Partie I :

Q1 : Type de canaux dans le simulateur :

Il ya deux type de canaux : des canaux en DownLink et des canaux et UpLink

Localisation du répertoire :



Q2 : Les paramètres associés a chaque type de canal :

Propriété des canaux UpLink :

```
properties
    % Feedback buffer. Implements the delay as a circular buffer of length N
    feedback_buffer
    % Current_index where the data was last written
    insert_index
    % Current_index from where to retrieve the data
    retrieve_index
    % Delay in TTIs
    feedback_delay

    % User to which this channel model is attached to
    attached_UE
end
```

Propriétés des canaux DownLink :

```
properties
    % These could actually be maps or an implementation that directly
    % calculates every time it is invoked.
    macroscopic_pathloss_model_is_set = false;
    macroscopic_pathloss_model
    shadow_fading_model_is_set = false;
    shadow_fading_model
    fast_fading_model_is_set = false;
    fast_fading_model

    % User to which this channel model is attached to
    attached_UE

    % This variables model the downlink signaling and the data that was
    % transmitted
    % RB_grid (retrieved via a function call)

end
```

Q3 : Définition du Shadowing et Fast-Fading : c'est des phénomènes aléatoires local qui donne une variation aléatoire a la puissance du signal parfois constructif et parfois destructif, elle rendent le calcul des frontières des cellule impossible et imprédictible. Ainsi le Shadowing est causé par l'atténuation des grand obstacles, quant au Fast-Fading a cause des réfraction du multi-trajet.

Dans le simulateur ils sont calculé ainsi :

Shadowing :

```
% Returns the shadow fading pathloss in dB between the given user's
% position and his eNodeB sector. Returns 0 in case no model is specified (for
% example, when Odyssey data would be used).
function [pathloss is_set] = shadow_fading_pathloss(obj)
    if ~obj.shadow_fading_model_is_set
        pathloss = 0;
        is_set = false;
        return
    else
        % Get eNodeB id and sector number
        eNodeB_id = obj.attached_UE.attached_site.id;
        pos = obj.attached_UE.pos;
        pathloss = obj.shadow_fading_model.get_pathloss(pos,eNodeB_id);
        is_set = true;
    end
end

% Returns the shadow fading pathloss in dB between the given user's
% position and a given eNodeB. Returns 0 in case no model is specified (for
% example, when Odyssey data would be used).
function [pathloss is_set] = interfering_shadow_fading_pathloss(obj,interferingSiteIds)
    if ~obj.shadow_fading_model_is_set
        pathloss = zeros([length(interferingSiteIds) 1]); % Column vector
        is_set = false;
        return
    else
        pos = obj.attached_UE.pos;
        pathloss = reshape(obj.shadow_fading_model.get_pathloss(pos,interferingSiteIds),[length(interferingSiteIds) 1]);
        is_set = true;
    end
end
```

Il y a deux fonctions, la première c'est pour calculer le shadowing local par rapport un a secteur d'un noeud et la deuxième c'est pour calculer le shadowing d'un noeud complet par rapport à leur position ce qui nous rends un vecteur et la variable is_set est mis à true.

Fast-Fading :

```

% Returns the equivalent fading parameters for a given UE and time.
% Both the signal's and the interferers' parameters are fetched at
% the same time to improve efficiency. i.e. The interfering trace
% from all sectors is fetched at the same time, not only the
% neighboring ones so as to be able to completely vectorize the
% fetching code.
function ff_loss = fast_fading_pathloss(obj,t,tx_mode)
    if ~obj.fast_fading_model_is_set
        error('Fast fading model must be set.');
```

Il est calculé dans cette fonction en générant un fast fading grace a la fonction generate_fast_fading()

Q : Type Winner Channel :

Code dans le simulateur : Le code des paramètres du winner channel est dans les fichier : winnerChannelFactory.m, LTE_load_params_dependant.m

Partie II :

Q1 : Événement lors d'un TTI :

D'après le fichier LTE_sim_main on peut voir que lors d'un TTI, on commence par initialiser l'horloge puis on la fait avancer a 1, on fait bouger les utilisateurs (si le keep_still est a true) puis on calcule le SINR et on prépare le CQI feedback, les noeuds reçoivent le feedback des utilisateurs l'ordonnanceur alloue les ressources aux utilisateurs, puis on refait la même chose pour les cas du non 0-delay, et à chaque TTI multiple de 10 on évalue et on estime le temps pour finir.

Q2 : feedback_channel_delay : C'est la valeur du délai pour que le transmetteur reçoit le feedback de l'émetteur.

Valeur par défaut : 3

<input type="checkbox"/> CQI_mapper	1x1 st
<input type="checkbox"/> feedback_channel_delay	3
<input checked="" type="checkbox"/> unquantized_CQI_feedback	0
<input type="checkbox"/> SINR averaging	1x1 st

Mettre ca valeur a 0 :

```

39 - LTE_config.trace_version = 'v1'; %
40 - LTE_config.feedback_channel_delay = 0;
41
```

Erreur lors du lancement de la simulation avec cette valeur de delay feedback.

Rôle des méthodes `link_quality_model` : La fonction calcule le SINR du récepteur qui est la métrique utilisé pour calculer la qualité de la liaison

```
% Calculates the receiver SINR, which is the metric used to measure link quality
function link_quality_model(obj,config)

% Get current time
t = obj.clock.time;

% Get map-dependant parameters for the current user
interfering_eNodeBs = obj.attached_eNodeB.neighbors_eNodeB;
user_macroscopic_pathloss = obj.downlink_channel.macroscopic_pathloss +
```

et `link_performance_model` : Évalue si les donnée du TB (Transport Block) sont bien transmis ce qui est la métrique qui évalue la bonne performance du canal.

```
% Evaluate whether this TB arrived correctly by using the data from
% the link quality model and feeding it to the link performance
% model (BLER curves)
function link_performance_model(obj)

% Get RB grid
% the_RB_grid = obj.RB_grid;
```

Q3 : Définir le SINR : C'est le rapport signal bruit interférence, Donc c'est le signal émis auquel on rajoute le bruit puis les interférence des antennes.

Relation entre SINR linéaire et SINR db : $\text{SINR}[\text{dB}] = 10 \cdot \log(\text{SINR}[\text{W}])$

```
% Calculation of the post-equalization symbols SINR
SINR_db = 10*log10(SINR_linear);
% SIR_db = 10*log10(SIR_linear);
```

Q4 : Pour le calcul du SINR_linear deux paramètre compte la puissance du signal et l'interférence plus le bruit.

```
end
SINR_linear = RX_power ./ Interference_plus_noise_power.';
```

Diagramme de dépendence :

```
SINR_db => SINR_linear => Puissance émise
                        /
                        => Bruit + interférence
```


Q5 : Rx_total est la puissance émise par l'antenne, elle est calculé en ajoutant a la puissance (Puissance des données, puissance des signalements) les différents Fading (Pathloss + Shadowing + Fast Fading).

```
% Get the RX power (power allocation) from the target eNodeB
TX_power_data = the_RB_grid.power_allocation';
TX_power_signaling = the_RB_grid.power_allocation_signaling';
RX_total = (TX_power_data+TX_power_signaling)./user_macroscopic_pathloss_linear./user_shadow_fading_loss_linear;
RX_total = reshape([RX_total; RX_total],1,[])/(2);
```

Q6 : L'allocation des puissance par le noeud aux utilisateur est homogène :

```
%% Scheduler options
LTE_config.scheduler = 'round robin'; % 'round robin'
LTE_config.scheduler_params.fairness = 0.5; % fairness for the
% LTE_config.scheduler_params.alpha = 1; % Additional
% LTE_config.scheduler_params.beta = 1;
LTE_config.power_allocation = 'homogeneous'; % 'right now no
```

Partie III :

Q1 : Débogage de la politique round robin et BestCQI :

Round robin : cette algorithme attribue des RessourcesBlock a chaque UE a tour de role (tourniquet) : on voit bien dans cette figure l'enchaînement des utilisateur 1 2 3 ... 10 donc il sélectionne via l'ordre de l'ID de l'utilisateur :

1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10
10	1
11	2
12	3
13	4
14	5
15	6
16	7
17	8
18	9
19	10
20	1

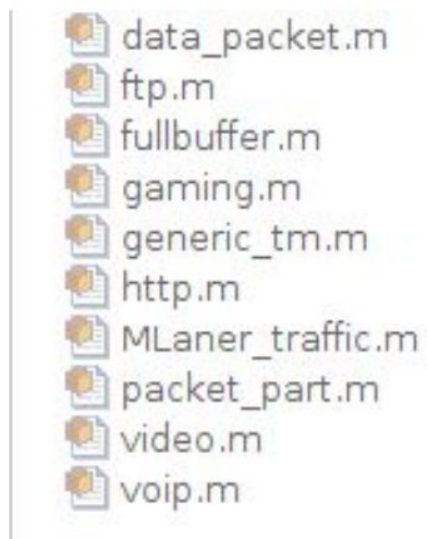
BestCQI : Cette algorithme attribue les RessourcesBlock à l'utilisateur qui a le meilleure valeur de CQI a chaque fois :

1	
2	9
3	9
4	9
5	9
6	9
7	9
8	9
9	9
10	9
11	9
12	9
13	9
14	9
15	9
16	9
17	9
18	9

Ainsi dans ce cas c'est l'utilisateur 9 qui est choisi a chaque fois car il a le meilleure CQI.

Q2 : Les types de trafic utilisés par le simulateur :

On peut voir dans le fichier +traffic_modelset les type de trafic suivant :



Q3 : Le rôle de la variable **aPrioriPdf** dans les différents types de trafic :

```
aPrioriPdf = [0.1,0.2,0.2,0.3,0.2];
```


Cette variable permet d'attribuer une priorité a des trafic précis par exemple ici nous avons attribuer : 10% au FTP; 20% au HTTP, video et à la VOIP et enfin 30% au gaming.

Q4 : Simulation qu'utilise tous les types de trafic :

pour faire ce genre de simulation tout d'abord il faut utilisé le type de scheduler à 'alpha fair' puis sélectionner ces variables :

```
LTE_config.traffic_models.usetraffic_model=true;  
LTE_config.traffic_models.type='3GPP';
```

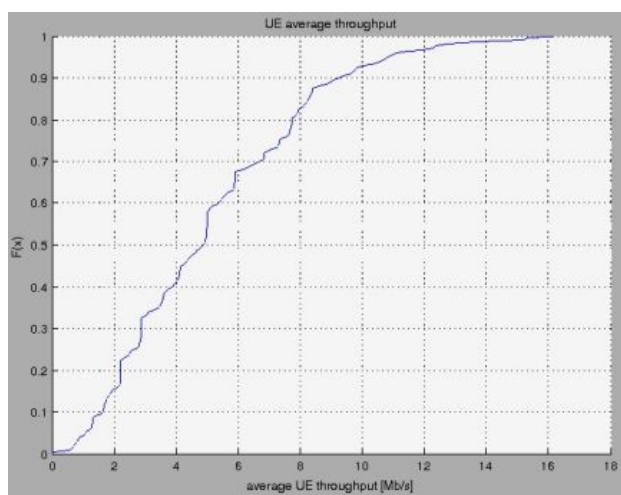
et ainsi lors de débogage nous pouvons voir les trafics des différent utilisateurs :

 traffic_model	1x1 traffic_models.voip
 traffic_model	1x1 traffic_models.http
 traffic_model	1x1 traffic_models.voip

Nombre d'utilisateurs de chaque type de trafic : 1

Q5 : Comparaison des deux débit de simulation :

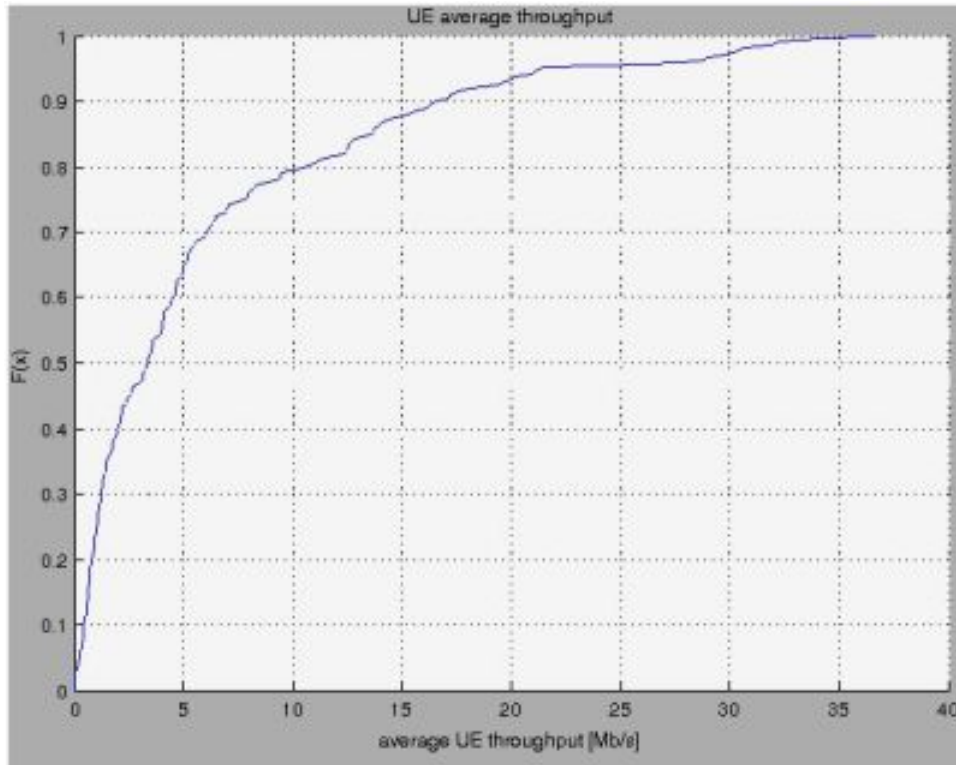
Pour une simulation en FullBuffer on sélectionne un scheduler Round Robin ainsi on aura le résultat :



avec un débit moyen de 5Mb/s

Pour le FTP, on ajoute une valeur a la fonction LTE_trafficmodel qui lui dit qu'on veut ajouter du FTP donc on choisi la valeur 1 et on met le scheduler au type 'alpha fair' :

```
LTE_trafficmodel(LTE_config.traffic_models,UEs(u_),max(LTE_config.feedback_channel_delay,0), 1);
```



On aura un débit moyen de 6Mb/s.

ainsi le FullBuffer a un débit moyen inférieur au débit sans Fullbuffer ce qui est incorrecte en théorie.