ΒΑΣΙΚΕΣ ΑΡΧΕΣ ΔΙΚΤΥΩΝ ΚΙΝΗΤΩΝ ΕΠΙΚΟΙΝΩΝΙΩΝ

Scheduling Αλγόριθμοι σε MATLAB για Downlink σε LTE-4G

ΒΟΥΛΓΑΡΗΣ ΝΙΚΟΛΑΟΣ, 1084626

ΑΘΑΝΑΣΙΟΣ ΣΠΥΡΙΔΩΝ ΣΚΑΝΔΑΛΟΣ, 1084646

Σύνδεσμος για GitHub Repository με τα αρχεία της εργασίας

Ο στόχος της εργασίας ήταν να σχεδιαστούν και αξιολογηθούν πέντε schedulers που χρησιμοποιούνται για LTE Downlink Transmission και να δοκιμαστεί η απόδοση του δικτύου σε διαφορετικά σενάρια. Επιλέξαμε να υλοποιήσουμε τους παρακάτω: Round Robin, Proportional Fair, Maximum Carrier-to-Interference, Packet Loss Ratio και Exponential Proportional Fair. Η αποτίμηση τους θα γίνει με χρήση ως ανεξάρτητης μεταβλητής (input) το πλήθος των χρηστών, και με χρήση εξαρτημένων μεταβλητών (outputs) τα throughput και fairness της λειτουργίας του δικτύου. Η υλοποίηση έγινε σε Windows 10 με MATLAB R2024b, με χρήση του 5G Toolbox.

1. Υλοποίηση της δικτυακής υποδομής

Σε αυτή την προσομοίωση, μοντελοποιούμε ένα κυψελοειδές δίκτυο με έναν μόνο σταθμό βάσης (gNB) και πολλαπλούς εξοπλισμούς χρηστών (UEs). Χρησιμοποιείται nrGNB object από το Toolbox για τη δημιουργία και διαμόρφωση κόμβου gNB. nrUE objects για τη δημιουργία και διαμόρφωση UE κόμβων και η συνάρτηση configureScheduler(gnb,Name=Value) για τη διαμόρφωση του scheduling αλγόριθμου.

Η συνάρτηση αυτή δίνεται από το toolbox και ορίζει τις παραμέτρους χρονοπρογραμματισμού χρησιμοποιώντας ένα ή περισσότερα προαιρετικά ορίσματα ονόματος-τιμής. Οι packet schedulers είναι υπεύθυνοι για την κατανομή των πόρων στα πακέτα των χρηστών και εφαρμόζονται στο Medium Access Layer (MAC) που φιλοξενείται στον gNB. Η configureScheduler συνάρτηση προσφέρει την επιλογή ανάμεσα σε τρεις υλοποιημένους scheduling αλγορίθμους, καθώς και την δυνατότητα για δημιουργία custom αλγορίθμου από τον χρήστη. Από τους υλοποιημένους κρατάμε τον Round Robin για σύγκριση με αυτούς που έχουμε υλοποιήσει εμείς.

Δημιουργούμε τον gNB και πολλαπλούς UE κόμβους σε τυχαίες συντεταγμένες γύρω από αυτόν. Ο αριθμός των UEs ορίζεται από τον χρήστη. Στον ορισμό του gNB καθορίζεται και ο αριθμός των διαθέσιμων Resource Blocks, ο οποίος εξαρτάται από τις τιμές των ChannelBandwidth και SubcarrierSpacing.

```
gNB = nrGNB(Position=[0 0 0],DuplexMode="FDD",CarrierFrequency=2.6e9, ...
    ChannelBandwidth= 20e6, SubcarrierSpacing= 30e3, ...
    NumResourceBlocks=20);

% Define the random range for UE positions around gNB
positionRange = 1000; % Set the range for x and y positions around the gNB

% Generate random positions around gNB with a fixed z-coordinate
uePositions = [positionRange * (rand(UEnum, 1) - 0.5), ...  % Random x positions around gNB within +/- 500 meters
               positionRange * (rand(UEnum, 1) - 0.5), ...  % Random y positions around gNB within +/- 500 meters
               zeros(UEnum, 1)];                        % Fixed z-coordinate for all UEs

% Generate unique names for each UE
ueNames = "UE-" + (1:UEnum);

% Create UE nodes with specified properties
UEs = nrUE(Name=ueNames, Position=uePositions);
```

Δημιουργία κόμβων

Καλείται η configureScheduler με schedulerType που ορίζει ο χρήστης, είτε built in είτε custom. Χρησιμοποιούμε την connectUE για να συνδέσουμε τον gNB με τους UEs. Προσθέτουμε διαφορετικό τύπο traffic (για κάθε πείραμα που θα κάνουμε στην πορεία) στη DL κατεύθυνση, προκειμένου να στέλνουμε συνεχώς πακέτα στους χρήστες. Προσθέτουμε όλους τους κόμβους, καθώς και ένα 3GPP TR 38.901 Channel Model, στην προσομοίωση. Το μοντέλο θέτει το περιβάλλον της προσομοίωσης σε Urban Macro (UMa) με τις εκτάσεις του να ορίζονται με βάση τις θέσεις των UE. Ένα μοντέλο καναλιού χαρακτηρίζει τον τρόπο διάδοσης των σημάτων από τον πομπό στον δέκτη, το οποίο είναι ζωτικής σημασίας για την αξιολόγηση των αλγορίθμων χρονοπρογραμματισμού.

```
% Configure a scheduler at the gNB
if ischar(schedulerType) || isstring(schedulerType)
    % Use the built-in scheduler specified by name
    configureScheduler(gNB, Scheduler=schedulerType, ResourceAllocationType=0);
else
    % Instantiate the custom scheduler
    customScheduler = schedulerType();
    configureScheduler(gNB, Scheduler=customScheduler, ResourceAllocationType=0);
end

% Connect the UE nodes to the gNB node.
connectUE(gNB,UEs,FullBufferTraffic="DL")

%{ ...

% Add the nodes to the network simulator.
addNodes(networkSimulator,gNB)
addNodes(networkSimulator,UEs)

% Use 3GPP TR 38.901 channel model for all links. You can also run the example with a free space path loss model.
%  Sets the propagation environment to "UMa" (Urban Macro), which is one of several deployment scenarios supported by 3GPP TR 38.901
%  ScenarioExtents: Specifies the simulation area boundaries in the x and y directions.
%  This ensures that the channel model accounts for distances and interactions between nodes within this area.
channelModel = "3GPP TR 38.901";

if strcmp(channelModel,"3GPP TR 38.901")
    % Define scenario boundaries
    pos = reshape([gNB.Position UEs.Position],3,[]);
    minX = min(pos(1,:));          % x-coordinate of the left edge of the scenario in meters
    minY = min(pos(2,:));          % y-coordinate of the bottom edge of the scenario in meters
    width = max(pos(1,:)) - minX;  % Width (right edge of the 2D scenario) in meters, given as maxX - minX
    height = max(pos(2,:)) - minY; % Height (top edge of the 2D scenario) in meters, given as maxY - minY

    % Create the channel model
    channel = h38901Channel(Scenario="UMa",ScenarioExtents=[minX minY width height]);
    % Add the channel model to the simulator
    addChannelModel(networkSimulator,@channel.channelFunction);
    connectNodes(channel,networkSimulator);
end
```

Ορισμός Scheduler, Κόμβων και Μοντέλου Καναλιού στη Προσομοίωση

Εκτελούμε το simulation και λαμβάνουμε τη δομή statistics για κάθε κόμβο. Στη μεταβλητή ueStats αποθηκεύουμε τις τιμές των statistics.MAC, που περιλαμβάνουν τιμές για τα

ReceivedBytes και χρησιμοποιούνται παρακάτω. Για την αξιολόγηση των αλγορίθμων χρησιμοποιούμε τις μετρικές με βάσει τους παρακάτω τύπους:

- Average Throughput, που είναι ο μέσος ρυθμός επιτυχούς μεταφοράς δεδομένων σε ένα κανάλι επικοινωνίας. Μετρά την αποδοτικότητα του δικτύου όσον αφορά τη χρήση των πόρων του. Υπολογίζεται διαιρώντας τα συνολικά δεδομένα (ReceivedBytes) που ελήφθησαν με το συνολικό χρόνο προσομοίωσης.

$$Throughput = \frac{\sum \text{Rx Packet Size}}{\text{Delivery Time}}$$

- Jain's Fairness Index, που είναι ο μετρητής για την ποσοτικοποίηση της δίκαιης κατανομής των πόρων μεταξύ των χρηστών. Βοηθά στην αξιολόγηση του πόσο ομοιόμορφα κατανέμονται οι πόροι μεταξύ των χρηστών. Έχει εύρος τιμών μεταξύ 0 και 1, όπου το 1 υποδηλώνει τέλεια δικαιοσύνη. Υπολογίζεται με το παρακάτω τύπο, όπου το x_i το throughput του $i_{στου}$ UE και n ο συνολικός αριθμός των UEs.

$$f(x) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \sum_{i=1}^{n} x_i^2}$$

Χρησιμοποιούμε τα ReceivedBytes μετατρέποντας σε bits και τα διαιρούμε με το simulationTime για να υπολογίσουμε το throughput για κάθε UE. Με βάσει αυτά υπολογίζουμε το Average Throughput, καθώς και το Jain's Fairness Index.

```
% Run the simulation for the specified number of frames numFrameSimulation.
% Calculate the simulation duration (in seconds)
simulationTime = numFrameSimulation*1e-2;

% Run the simulation
run(networkSimulator,simulationTime);

% Initialize an array to hold throughput values for each UE
throughputUEs = zeros(UEnum,1);


% Loop through each UE and calculate throughput
for i = 1:UEnum
    % Get statistics for each UE
    ueStats = statistics(UEs(i));
    macStats = ueStats.MAC;

    % Display the MAC statistics for each UE
    fprintf("MAC Statistics for %s:\n", ueNames(i));
    disp(macStats);

    % Extract the total received bytes from the MAC statistics
    if isfield(macStats, 'ReceivedBytes')
        totalReceivedBits = macStats.ReceivedBytes * 8; % Convert bytes to bits
    else
        error('The field "ReceivedBytes" was not found in the MAC statistics.');
    end

    %global transmittedx;
    %global retransmittedx;
    %transmittedx(i) = macStats.DLTransmissionRB;
    %retransmittedx(i) = macStats.DLRetransmissionRB;

    % Calculate throughput in bits per second
    throughputUEs(i) = totalReceivedBits / simulationTime;
```

Υπολογισμός Throughput και Fairness Μετρικών

Ότι έχουμε αναφέρει παραπάνω εμπεριέχεται μέσα σε μια simulationRun(UEnum, schedulerType) συνάρτηση που καλείται για κάθε scheduling αλγόριθμο Η παραπάνω διαδικασία γίνεται για κάθε scheduling algorithm και διαφορετικό αριθμό UEs. Στο iteration_num αποθηκεύονται πολλαπλάσια του 4 με βάσει τον αριθμό που εισάγει ο χρήστης και χρησιμοποιούνται για τον ορισμό του αριθμού των UEs. Τα δεδομένα για το Average Throughput και το Jain's Fairness Index αποθηκεύονται σε array και εκτυπώνονται σε δυο plot για σύγκριση αποτελεσμάτων.

2. Scheduling Αλγόριθμοι

Round Robin

Ο Round Robin scheduling αλγόριθμος κατανέμει ίσα μπλοκ χρόνου ή πόρων σε κάθε UE με κυκλική σειρά. Με αυτό τον τρόπο εξασφαλίζει δικαιοσύνη όσον αφορά την κατανομή του χρόνου. Υπάρχει υλοποιημένος στη configureScheduler και επιλέξαμε να μην ετοιμάσουμε custom κώδικα. Οι υπόλοιποι αλγόριθμοι υλοποιήθηκαν από εμάς.

Proportional Fair

Ο Proportional Fair scheduler υπάρχει επίσης υλοποιημένος στη configureScheduler, αλλά επιλέξαμε να ετοιμάσουμε ένα custom κώδικα για καλύτερη σύγκριση με τους υπόλοιπους που ετοιμάσαμε. Πρόκειται για ένα αλγόριθμο που βασίζεται στη διατήρηση μιας ισορροπίας μεταξύ δύο ανταγωνιστικών συμφερόντων. Προσπαθώντας να μεγιστοποιήσει το συνολικό throughput του δικτύου, ενώ ταυτόχρονα επιτρέπει σε όλους τους χρήστες ένα ελάχιστο επίπεδο εξυπηρέτησης. Αυτό επιτυγχάνεται αναθέτοντας σε κάθε ροή δεδομένων μια προτεραιότητα scheduling που είναι αντιστρόφως ανάλογη της αναμενόμενης κατανάλωσης πόρων της.

$$\text{PF Metric (User i)} = \frac{\text{Instantaneous Data Rate (User i)}}{\text{Average Throughput (User i)}}$$

Αυτή η μετρική εξασφαλίζει ότι οι χρήστες με καλές στιγμιαίες συνθήκες καναλιού (υψηλό data rate) και χαμηλό average throughput έχουν προτεραιότητα, γεγονός που εξισορροπεί τη δικαιοσύνη και την αποδοτικότητα του συστήματος.

Θα εξηγήσουμε τώρα την δομή των script που έχουμε χρησιμοποιήσει για τους αλγόριθμους. Είναι ίδια για όλους με μικρές διαφορές που θα εξηγηθούν στην πορεία. Όλοι οι αλγόριθμοι ορίζονται ως μια νέα κλάση που κληρονομεί από την nrScheduler, η οποία είναι η βασική κλάση για τους scheduling αλγόριθμους σε έναν προσομοιωτή 5G NR. Διατηρούμε ένα map AverageThroughput για την αποθήκευση των ιστορικών throughput κάθε UE, τα οποία αναγνωρίζονται από το RNTI τους (Radio Network Temporary Identifier). Στο σύνολο του κώδικα χρησιμοποιείται η συνάρτηση scheduleNewTransmissionsDL(newradioscheduler, timeResource, frequencyResource, schedulingInfo) του toolbox, η οποία αναθέσει πόρους στους UE για νέες downlink transmissions σε ένα χρονικό διάστημα μετάδοσης (TTI).

Ελέγχουμε αρχικά εάν υπάρχουν UE που είναι υποψήφια για scheduling. Αξιοποιούμε τη παράμετρο schedulingInfo της παραπάνω συνάρτησης, μια δομή που περιέχει τις πληροφορίες για τις αποφάσεις scheduling εντός μιας TTI. Περιέχει ένα array EligibleUEs με τα RNTIs των UE που είναι υποψήφιοι για κατανομή στο τρέχον TTI. Αυτό το σύνολο περιλαμβάνει όλους τους κόμβους UE που συνδέονται με τον gNB, εκτός από εκείνους που πληρούν κάποιο από τα εξής κριτήρια: Έχουν retransmission σε αυτό το TTI ή δεν έχουν δεδομένα στο queue.

Στη συνέχεια βρίσκουμε τα διαθέσιμα Resource Blocks. Έχουμε αναφέρει ότι έχει προστεθεί ένα 3GPP TR 38.901 Channel Model στην προσομοίωση. Εφόσον έχουμε ορίσει στη configureScheduler το ResourceAllocationType σε 0, μπορούμε να χρησιμοποιήσουμε την παράμετρο frequencyResponse. Είναι ένας πίνακας bit με μήκος ίσο με τον αριθμό των Resource Block Groups (RBGs) στο DL bandwidth. Κάθε bit αντιπροσωπεύει ένα RBG. Το 0 υποδεικνύει ότι το αντίστοιχο RBG είναι διαθέσιμο για νέο προγραμματισμό μετάδοσης, ενώ το 1 υποδεικνύει ότι δεν είναι. Για τον προσδιορισμό του αριθμού block σε μια ομάδα, η κλάση nrScheduler χρησιμοποιεί τη παρακάτω διαμόρφωση, με βάσει την προδιαγραφή 3GPP.

Number of NRBs	RBG Size
1–36	2
37–72	4
73–144	8
145–275	16

Προσδιορισμός Αριθμού Resource Block σε ένα Group

```
% Get eligible UEs and available RBGs
eligibleUEs = schedulingInfo.EligibleUEs;
numEligibleUEs = numel(eligibleUEs); % Store number of elements
if numEligibleUEs == 0
    dlAssignments = []; % No UEs to schedule
    return;
end
disp("Eligible UEs:");
disp(eligibleUEs);


% Find available RBGs
availableRBGs = find(frequencyResource == 0); % Return array
numAvailableRBGs = numel(availableRBGs);
disp(['Total RBGs: ', num2str(length(frequencyResource))]);
disp(['Number of Available RBGs: ', num2str(numAvailableRBGs)]);
disp(frequencyResource);

if numAvailableRBGs == 0
    dlAssignments = []; % No resources to allocate
    return;
end

% Initialize arrays to store PF metrics
PFmetric = zeros(1, numEligibleUEs);
estimatedDataRate = zeros(1, numEligibleUEs);
mcsIndices = zeros(1, numEligibleUEs);
spectralEfficiencies = zeros(1, numEligibleUEs);

% For each eligible UE, compute PF metric
for i = 1:numEligibleUEs
    ueRNTI = eligibleUEs(i);

    % Existing UEs retrieve historic average throughput
    if isKey(obj.AverageThroughput, ueRNTI)
        avgThroughput = obj.AverageThroughput(ueRNTI);
    else
        avgThroughput = 1; % You can adjust this initial value
    end
```

Προσδιορισμός Υποψήφιων UEs και Ιστορικού Average Throughput

Τα νέα UE ξεκινούν με μια default throughput, τα υπάρχοντα χρησιμοποιούν την παρελθοντική τους average throughput. Μετά από πειραματισμούς και λόγω δημιουργίας προβλημάτων στους υπολογισμούς των PF μετρικών με μεγάλο αριθμό UE, επιλέξαμε η default τιμή που ξεκινάνε τα νέα UE να είναι 1. Η τιμή του instantaneous data rate αποθηκεύεται στην μεταβλητή estimatedDataRate και υπολογίζεται με βάσει την ποιότητα του καναλιού. Στις επόμενες παραγράφους κάνουμε μια σύντομη αναφορά στις δυο παραμέτρους που αντιπροσωπεύουν την ποιότητα του καναλιού.

Πρώτη παράμετρος είναι ο δείκτης MCS. Υψηλότεροι δείκτες MCS αντικατοπτρίζουν καλύτερες συνθήκες καναλιού. Σε κανονικές συνθήκες, ο δείκτης επιλέγεται δυναμικά με το CQI (Δείκτης ποιότητας καναλιού). Επιχειρήσαμε να το υλοποιήσουμε αυτό, αλλά για λόγους απλότητας και αποτελεσματικότητας των αλγορίθμων, επιλέξαμε να το δώσουμε μια τυχαία τιμή μέχρι 27, σύμφωνα με τις προδιαγραφές του 3GPP. Δεύτερη παράμετρος είναι η spectral efficiency. Δηλώσει το πόσο αποτελεσματικά ένα channel χρησιμοποιεί το διαθέσιμο bandwidth για τη μετάδοση δεδομένων. Κάθε MCS δείκτης αντιστοιχεί σε μια συγκεκριμένη spectral efficiency με βάση το σχήμα διαμόρφωσης. Οι υψηλότεροι δείκτες MCS έχουν υψηλότερη spectral efficiency επειδή χρησιμοποιούν σχήματα διαμόρφωσης υψηλότερης τάξης και υψηλότερους ρυθμούς κωδικοποίησης.

Στο script, ένας mcsIndex εκχωρείται σε κάθε UE χρησιμοποιώντας τη συνάρτηση estimateMCSIndex. Ένα UE με υψηλό mcsIndex επιτρέπει υψηλούς instantaneous data rates. Η συνάρτηση getSpectralEfficiency αντιστοιχίζει τον δείκτη MCS στην αντίστοιχη τιμή spectral efficiency χρησιμοποιώντας έναν προκαθορισμένο πίνακα. Η τελική τιμή αποθηκεύεται στη estimatedDataRate. Έχουμε πλέον ότι χρειαζόμαστε για να υπολογίσουμε τη μετρική PF για κάθε UE.

```
% Estimate instantaneous data rate based on channel conditions
% For simplicity, we'll assign random MCS indices
ueContext = obj.UEContext(ueRNTI);
mcsIndex = obj.estimateMCSIndex(ueContext);
mcsIndices(i) = mcsIndex;

% Get the spectral efficiency from MCS index
spectralEfficiency = obj.getSpectralEfficiency(mcsIndex);
spectralEfficiencies(i) = spectralEfficiency;

% Estimate instantaneous data rate (assuming one RBG)
estimatedDataRate(i) = spectralEfficiency * 1; % Multiply by number of RBGs (1)

% Compute PF metric
% Higher metrics prioritize UEs with high data rates relative to their average throughput.
PFmetric(i) = (estimatedDataRate(i) / avgThroughput);
```

Υπολογισμός Instantaneous Data Rate και PF μετρικής

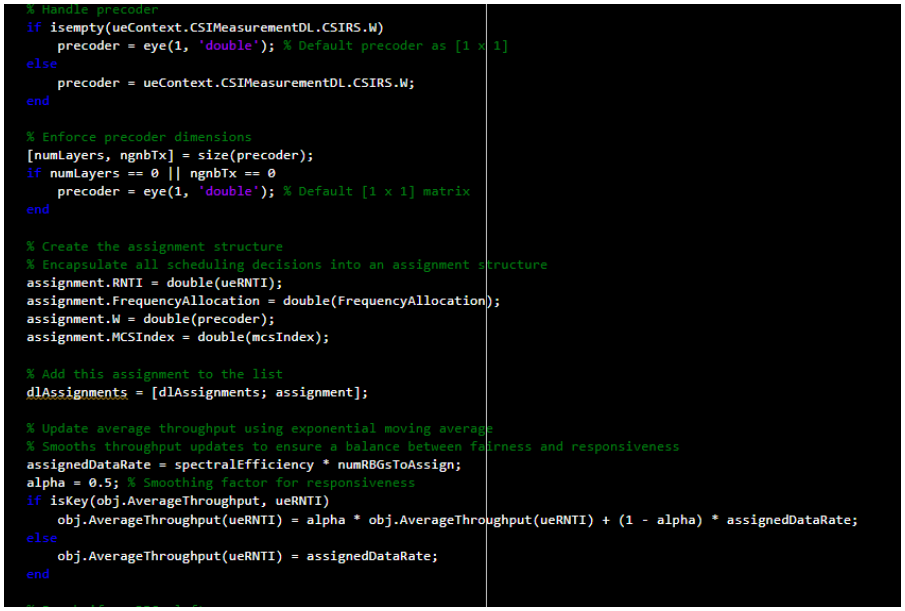
Αρχικοποιείται μεταβλητή currentRBGIndex για παρακολούθηση των blocks που έχουν χρησιμοποιηθεί. Υπολογίζουμε την μερίδα πόρων που δικαιούται να πάρει ο UE, αναλογικά με την μετρική του σε σχέση με αυτή των υπολοίπων. Ελέγχουμε αν υπάρχουν αρκετοί διαθέσιμοι πόροι και αν πράγματι υπάρχουν, μέσω του assignedRBGs αναθέτουμε τα RBGs σε κάθε UE, με σεβασμό στους περιορισμούς. Αυξάνουμε την τιμή του currentRBGIndex και ανανεώνουμε το Frequency Allocation για το συγκεκριμένο UE προκειμένου να φαίνεται πόσα RBG του έχουν ανατεθεί.

```
% Loop through sorted UEs and assign resources
for idx = 1:numEligibleUEs
    disp(["Position in sorted array: ", num2str(idx)]);

    ueRNTI = sortedUEs(idx);
    disp(["RNTI :", num2str(ueRNTI)]);

    spectralEfficiency = sortedSpectralEfficiencies(idx);
    mcsIndex = sortedMCSIndices(idx);

    % Determine number of RBGs to assign proportionally
    % UEs with higher PF metrics get a proportionally larger share of RBGs
    proportion = PFmetric(sortedIndices(idx)) / sum(PFmetric(sortedIndices(idx:end)));
    numRBGsToAssign = max(floor(proportion * totalRemainingRBGs), 1); % At least 1 RBG

    % Check if enough RBGs are left
    if currentRBGIndex + numRBGsToAssign - 1 > numAvailableRBGs
        numRBGsToAssign = numAvailableRBGs - currentRBGIndex + 1;
    end

    % Assign RBGs to this UE respecting resource constraints
    assignedRBGs = availableRBGs(currentRBGIndex : currentRBGIndex + numRBGsToAssign - 1);
    disp(["Assigned :", num2str(assignedRBGs)]);
    currentRBGIndex = currentRBGIndex + numRBGsToAssign;
    totalRemainingRBGs = totalRemainingRBGs - numRBGsToAssign;

    % Update frequency allocation for this UE
    FrequencyAllocation = zeros(1, length(frequencyResource), 'double');
    FrequencyAllocation(assignedRBGs) = 1;
    % Get UE context for precoder
    ueContext = obj.UEContext(ueRNTI);
```

Ανάθεση Resource Block

Η μεταβλητή precoder αντιπροσωπεύει τον πίνακα precoding W, ο οποίος προέρχεται από την παράμετρο CSIMeasurementDL του nrScheduler. Ο W έχει σχεδιαστεί για να μεγιστοποιεί την ποιότητα του σήματος και της μετάδοσης στο δέκτη με βάση το channel matrix. Η ύπαρξη του στο κώδικα είναι απαραίτητη για προηγμένες τεχνικές όπως η διαμόρφωση δέσμης ή η χωρική πολυπλεξία, επιτρέποντας την ταυτόχρονη μετάδοση πολλαπλών επιπέδων.

Έχοντας πάρει όλες τις αποφάσεις για το scheduling, δημιουργούμε μια μεταβλητή assignment για το UE, η οποία τις ενθυλακώνει σε ένα structure για προσθήκη στη dlAssignments λίστα. Τέλος ανανεώνουμε την τιμή του average throughput για το UE βάσει του εκθετικού κινητού μέσου όρου. Πρόκειται για ένα τύπο που δίνει μεγαλύτερη βαρύτητα και σημασία στα πιο πρόσφατα data rates από ότι στην προηγούμενη τιμή του average throughput. Χρησιμοποιεί μια μεταβλητή alpha ως παράγοντα εξομάλυνσης των νέων τιμών του throughput, προκειμένου να εξασφαλίσει ισορροπία μεταξύ δικαιοσύνης και απόκρισης.



```
% Handle precoder
if isempty(ueContext.CSIMeasurementDL.CSIRS.W)
    precoder = eye(1, 'double'); % Default precoder as [1 x 1]
else
    precoder = ueContext.CSIMeasurementDL.CSIRS.W;
end

% Enforce precoder dimensions
[numLayers, ngnbTx] = size(precoder);
if numLayers == 0 || ngnbTx == 0
    precoder = eye(1, 'double'); % Default [1 x 1] matrix
end

% Create the assignment structure
% Encapsulate all scheduling decisions into an assignment structure
assignment.RNTI = double(ueRNTI);
assignment.FrequencyAllocation = double(FrequencyAllocation);
assignment.W = double(precoder);
assignment.MCSIndex = double(mcsIndex);

% Add this assignment to the list
dlAssignments = [dlAssignments; assignment];

% Update average throughput using exponential moving average
% Smooths throughput updates to ensure a balance between fairness and responsiveness
assignedDataRate = spectralEfficiency * numRBGsToAssign;
alpha = 0.5; % Smoothing factor for responsiveness
if isKey(obj.AverageThroughput, ueRNTI)
    obj.AverageThroughput(ueRNTI) = alpha * obj.AverageThroughput(ueRNTI) + (1 - alpha) * assignedDataRate;
else
    obj.AverageThroughput(ueRNTI) = assignedDataRate;
end
```

Διαχείριση Precoder και Ανανέωση τιμής Throughput

Exponential Proportional Fair

Η δομή του script για τον Exponential Proportional Fair αλγόριθμο, όπως αναφέρθηκε παραπάνω, είναι ίδια με αυτή του Proportional Fair. Διαφορά έχουν μόνο στην μετρική που χρησιμοποιούν για την προτεραιότητα. Στο Exponential PF ισούται με $Ri * e^{-\beta \cdot Ti}$, όπου Ri είναι το instantaneous data rate, Ti είναι το average throughput και β το βάρος δικαιοσύνης. Η παράμετρος β παίζει κρίσιμο ρόλο στην εξισορρόπηση του συμβιβασμού μεταξύ fairness και throughput. Η τιμή της επηρεάζει άμεσα τον τρόπο με τον οποίο κατανέμονται οι πόροι, ελέγχοντας πόσο επιθετικά ο scheduler δίνει προτεραιότητα στο fairness.

1.Χαμηλό β(β≈0)

Το exponential weighting γίνεται αμελητέο, οπότε συμπεριφέρεται σχεδόν σαν scheduler μεγιστοποίησης throughput (max-rate scheduler). Δίνει προτεραιότητα στα UEs με τους υψηλότερους instantaneous data rates. Αγνοεί σημαντικά τη δικαιοσύνη, οδηγώντας σε starvation τα UEs με κακές συνθήκες καναλιού.

2. Μέτριο β(β≈0,5)

Εξισορροπεί την αποδοτικότητα των δυο μετρικών. Τα UEs με χαμηλή average throughput επιλέγονται περισσότερο σε σύγκριση με εκείνα με υψηλή, αλλά το data rate εξακολουθεί να παίζει σημαντικό ρόλο.

3. Υψηλό β(π.χ., β>1)

Η fairness κυριαρχεί και ο scheduler δίνει μεγάλη προτεραιότητα στα UE με χαμηλή average throughput, ακόμη και αν έχουν κακές στιγμιαίες συνθήκες καναλιού. Αυξάνει τη δικαιοσύνη, αλλά μπορεί να υποβαθμίσει σημαντικά τη συνολική απόδοση του συστήματος. Κάποιοι UEs λαμβάνουν

πόρους και αν δεν μπορούν να τους χρησιμοποιήσουν αποτελεσματικά λόγω κακής ποιότητας καναλιού.

β Value	Prioritization	Throughput	Fairness
$\beta = 0$	Max-rate scheduling (instantaneous R_i)	High	Low
$\beta = 0.5$	Balanced (mix of R_i and T_i)	Moderate	Moderate
$\beta > 1$	Strong fairness (favoring low T_i)	Low	High

Τιμές β και Αναμενόμενα Αποτελέσματα

Επειδή παρατηρούσαμε προβλήματα στα αποτελέσματα και για να αποτρέψουμε τον εκθετικό όρο από το να γίνει πολύ μικρός, εισάγουμε ένα ελάχιστο κατώφλι $e^{-\beta \cdot Ti} \geq \epsilon$. Αυτό διασφαλίζει ότι ο εκθετικός όρος αποφεύγει μετρικές κοντά στο μηδέν.

```
% Compute Exponential PF metric
% Higher metrics prioritize UEs with high data rates relative to their average throughput.
beta = 2; % Weighting factor for fairness  0.1 0.5 2 δοκιμασε
epsilon = 1e-3; % Minimum value for the exponential term
expTerm = exp(-beta * avgThroughput);
expTerm = max(expTerm, epsilon); % Ensure the term is not too small
EPFmetric(i) = estimatedDataRate(i) * expTerm;
```

Υπολογισμός EPF Μετρικής

Max Carrier-to-Interference (C/I)

Η δομή του script για τον Max Carrier-to-Interference (C/I) αλγόριθμο είναι ξανά ίδια με αυτή του Proportional Fair με διαφορά στην μετρική που χρησιμοποιούν για την προτεραιότητα και στον τρόπο υπολογισμού της ποιότητας καναλιού. Πρόκειται για έναν αλγόριθμο που έχει σχεδιαστεί για τη μεγιστοποίηση του throughput του συστήματος, με την απονομή πόρων σε χρήστες με την υψηλότερη instantaneous channel quality. Η επιλογή γίνεται με βάση το Carrier-to-Interference (C/I) ή Signal-to-Interference-plus-Noise Ratio (SINR), το οποίο υποδεικνύει αν ο χρήστης μπορεί να επιτύχει υψηλούς data rates.

Συλλέγουμε τιμές για τον δείκτη ποιότητας καναλιού (CQI) του κάθε UE από την μεταβλητή ueContext.CSIMeasurementDL.CSIRS.CQI του nrScheduler. Ο CQI αντικατοπτρίζει τις τρέχουσες συνθήκες καναλιού του UE, συνήθως σε κλίμακα από 1 έως 16. Υψηλότερες τιμές CQI υποδηλώνουν καλύτερη ποιότητα καναλιού. Βρίσκουμε τις τιμές spectral efficiency για το αντίστοιχο CQI και τις διαιρούμε με το interference για να υπολογίσουμε τις τιμές C/I.

Όσο αφορά το interference, σε Urban Macro περιβάλλον διακυμαίνεται από -90 έως -70 dB. Θέλουμε να είναι ίδια μορφή με την spectral efficiency για να πράξει διαίρεση. Επομένως μετατρέπουμε από db σε W με τον τύπο W = $10^{\frac{dBm-30}{10}}$ και από W σε bps/Hz χρησιμοποιώντας τον τύπο Shannon log(1+$\frac{interferanceW}{20MHz}$). Οι πράξεις αυτές προσεγγίζουν το interference που θα υπήρχε σε κανονικές συνθήκες.

```
function interference = estimateInterference(obj)
    % Generate random interference between -90 and -70 dBm
    interference_dBm = -90 + (rand() * (-70 - (-90))); % Random value in dBm
    interference_W = 10^((interference_dBm - 30) / 10); % Convert dBm to W

    % Convert interference to bps/Hz using Shannon Capacity formula
    bandwidth_Hz = 20e6; % Assume 1 MHz bandwidth for simplicity
    interference = log2(1 + interference_W / bandwidth_Hz); % Spectral efficiency (bps/Hz)
end

function spectralEfficiency = getSpectralEfficiencyFromCQI(obj, cqi)
    % Map CQI to spectral efficiency
    % CQI values range from 0 to 15, corresponding to different MCS levels
    cqiToSpectralEfficiency = [
        0.1523, 0.2344, 0.3770, 0.6016, 0.8770, 1.1758, 1.4766, ...
        1.9141, 2.4063, 2.7305, 3.3223, 3.9023, 4.5234, 5.1152, ...
        5.5547, 6.2266
    ];
    spectralEfficiency = cqiToSpectralEfficiency(max(1, min(cqi + 1, 16))); % Map CQI to efficiency
end
```

Καθορισμός Ποιότητας Καναλιού και Interference

Αξιοποιούνται οι πόροι από UEs που μπορούν να μεταδίδουν σε υψηλότερους data rates, καθώς η spectral efficiency τους είναι υψηλή. Η αποτελεσματική χρήση του φάσματος μεγιστοποιεί το throughput του συστήματος. Όσο αφορά την δικαιοσύνη όμως, τα UEs σε κακές συνθήκες καναλιού μπορεί να φτάσουν σε starvation, καθώς έχουν σταθερά χαμηλές τιμές C/I, με αποτέλεσμα χαμηλές τιμές fairness index.

```
    % Use reported CQI for channel quality
    cqi = ueContext.CSIMeasurementDL.CSIRS.CQI; % Assume CQI is directly accessible
    channelQuality = obj.getSpectralEfficiencyFromCQI(cqi);

    % Estimate interference (in bps/Hz)
    interference = obj.estimateInterference();

    % Compute C/I ratio
    CIratio(i) = channelQuality / max(interference, 1e-6); % Avoid division by zero

    % Estimate MCS index and data rate
    mcsIndex = obj.estimateMCSIndexFromCQI(cqi);
    mcsIndices(i) = mcsIndex;
    estimatedDataRate(i) = CIratio(i) * 1; % Assuming one RBG
end
```

Υπολογισμός C/I Μετρικής

Packet Loss Ratio

Το Packet Loss Ratio είναι αντιστρόφως ανάλογος της απόδοσης του δικτύου. Ορίζεται ως το κλάσμα του συνολικού αριθμού των πακέτων που αποστέλλονται από τον αποστολέα αλλά δεν λαμβάνονται στον προορισμό. Λόγω του ότι η MATLAB δεν μας παρέχει τρόπο να το παρακολουθούμε κατά τη διάρκεια της προσομοίωσης – και δεδομένου ότι η προσομοίωση μεταβάλλεται μόνο ως προς τον χρόνο με σταθερή κίνηση (traffic) – προχωρούμε στην εξής παραδοχή: σε κάθε simulation run, θεωρούμε το packet loss που υπολογίζουμε από το Proportional Fair scheduling ως αντιπροσωπευτικό για ολόκληρο το run.

Για να το πετύχουμε αυτό, δημιουργούμε τρεις πίνακες (arrays) μεγέθους ίσου με τον αριθμό των UE. Στους global πίνακες transmitted και retransmitted αποθηκεύουμε τα δεδομένα από τα

macStats.DLTransmissionRB και macStats.DLTransmissionRB αντίστοιχα των transmitted και retransmitted resource blocks. Στο global πίνακα packetloss αποθηκεύουμε τον λόγο των transmitted/retransmitted.

Η δομή του script CustomPacketLossScheduler είναι ίδια ξανά με τα υπόλοιπα με τις παρακάτω αλλαγές. Για μετρική προτεραιότητας χρησιμοποιούμε τις τιμές του packetloss πίνακα για να υπολογίσουμε 1/ PLS. Τα UEs ταξινομούνται και επιλέγονται πρώτα. Υπολογίζουμε για κάθε UE την τιμή 1−packetLoss/∑(packetLoss) και την αποθηκεύουμε στη μεταβλητή proportion. Αυτή η μεταβλητή χρησιμοποιείται για να κάνουμε κατανομή πόρων στα RBGs.

```
% Initialize arrays to store PLS metrics
% For each eligible UE, compute PLS metric
packet = zeros(1, numEligibleUEs);
for i = 1:numEligibleUEs
    ueRNTI = eligibleUEs(i);
    packet(i) = 1/packetloss(ueRNTI);
end

% Sort UEs based on PLS metric
[~, sortedIndices] = sort(packet, 'descend');
sortedUEs = eligibleUEs(sortedIndices);

% Allocate resources to UEs in order of PF metric
dlAssignments = [];
currentRBGIndex = 1;
totalRemainingRBGs = numAvailableRBGs;
% Loop through sorted UEs and assign resources
for idx = 1:numEligibleUEs
    ueRNTI = sortedUEs(idx);

    % Determine proportion of RBGs to assign
    proportion = packet(sortedIndices(idx)) / sum(packet(sortedIndices(idx:end)));
    numRBGsToAssign = max(floor(proportion * totalRemainingRBGs), 1); % At least 1 RBG

    % Check if enough RBGs are left
    if currentRBGIndex + numRBGsToAssign - 1 > numAvailableRBGs
        numRBGsToAssign = numAvailableRBGs - currentRBGIndex + 1;
    end
```

Κατανομή RB με βάσει το Packet Loss Ratio

3. Πειράματα και Συγκρίσεις

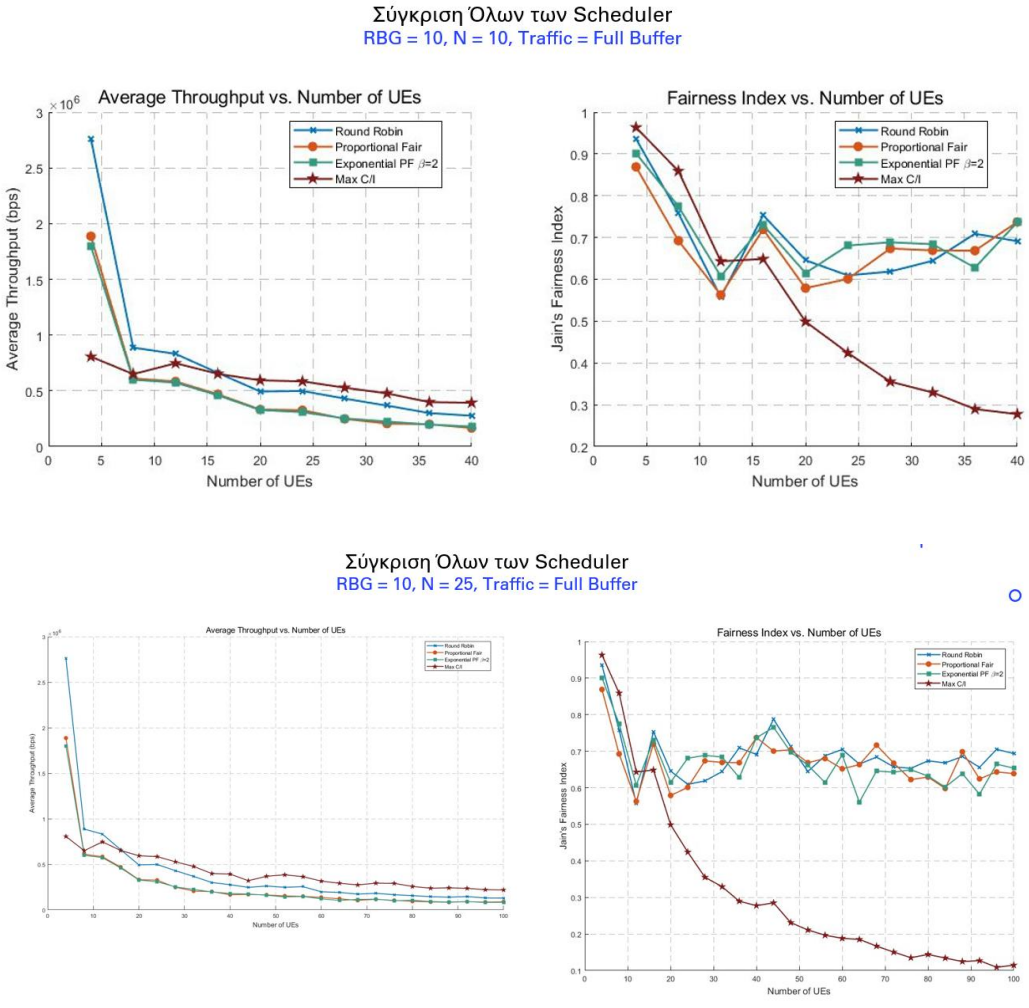
Για το πρώτο κομμάτι των πειραμάτων χρησιμοποιήθηκε Full Buffer Traffic ανάμεσα στη βάση και τα UEs. Πρόκειται για ένα μοντέλο traffic που χρησιμοποιείται σε προσομοιώσεις και δοκιμές δικτύων, στο οποίο οι UEs έχουν μόνιμα δεδομένα διαθέσιμα για μετάδοση ή λήψη. Εξασφαλίζει σταθερή ζήτηση για πόρους δικτύου. Πρόκειται για το worst case scenario για τον ανταγωνισμό πόρων και network load, όπου το δίκτυο λειτουργεί στη μέγιστη χωρητικότητά του. Το μοντέλο ακολουθεί ντετερμινιστική κατανομή για την παραγωγή δεδομένων, εξασφαλίζοντας ένα σταθερό φορτίο κίνησης. Το χρησιμοποιήσαμε αρχικά για να μελετήσουμε την αποτελεσματικότητα των scheduling αλγορίθμων, χωρίς παρεμβολές από τη μεταβλητότητα που φέρει η συμπεριφορά των πραγματικών traffic μοτίβων. Η προσομοίωση τρέχει με βάσει frames των 10ms.

Επειδή τα UEs με το μικρότερο packet loss θα δρομολογούνται πάντοτε πρώτα, ο αλγόριθμος παρουσιάζει πολύ κακές επιδόσεις όταν το traffic είναι Full Buffer. Οι τιμές ήταν τόσο χαμηλές που

επιλέξαμε να μην το συμπεριλάβουμε στις συγκρίσεις για αυτό το μοντέλο traffic. Θα ασχοληθούμε με αυτόν παρακάτω.

Δοκιμάζουμε αρχικά τους αλγορίθμους με διαθέσιμα 10 RBGs.

Scheduler	Iterations	Average Throughput	Fairness Index
Round Robin	10	7.5062e+05	0.6923
Proportional Fair	10	5.0265e+05	0.6773
Exponential PF, β=0.1	10	5.0311e+05	0.6864
Exponential PF, β=0.5	10	4.8072e+05	0.6227
Exponential PF, β=2	10	4.9211e+05	0.7046
Max C/I	10	5.8309e+05	0.5289
Round Robin	**25**	4.1104e+05	0.6699
Proportional Fair	**25**	2.7115e+05	0.6672
Exponential PF, β=2	**25**	2.6567e+05	0.6881
Max C/I	**25**	4.0596e+05	0.3117



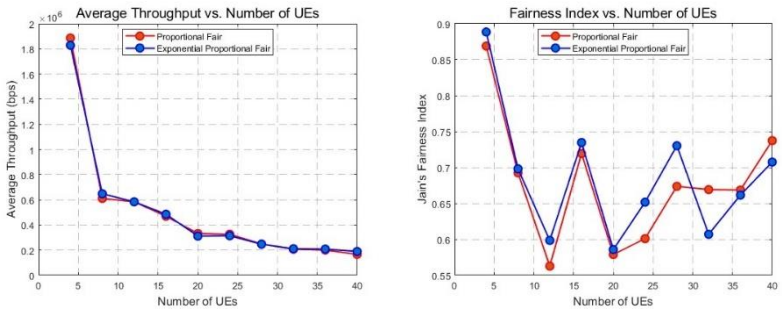
Σύγκριση Όλων των Scheduler με Full Buffer Traffic

Ο Round Robin scheduler, όπως αναφέρθηκε, κατανέμει πόρους κυκλικά σε όλους τους χρήστες χωρίς να λαμβάνει υπόψη τις συνθήκες του καναλιού τους ή το ιστορικό throughput. Με αυτό τον τρόπο, κάθε χρήστης λαμβάνει δίκαιο μερίδιο πόρων, ανεξάρτητα από την ποιότητα του καναλιού του. Στο σενάριο μας, όπου όλοι οι χρήστες έχουν παρόμοιες ποιότητες καναλιού, αυτό μπορεί να οδηγήσει σε συνεπές και προβλέψιμο throughput μεταξύ των χρηστών.

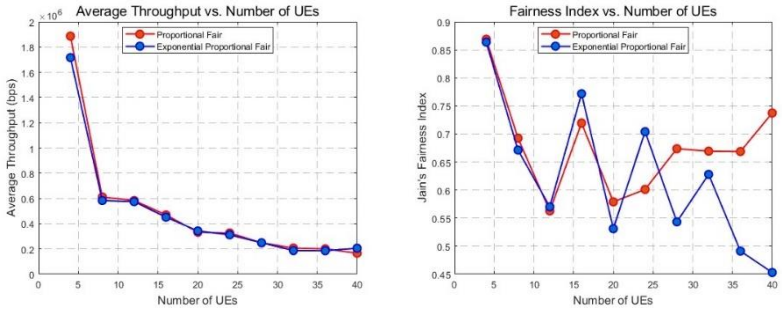
Ο Proportional Fair από την άλλη τιμωρεί τους χρήστες με υψηλές τιμές ιστορικού throughput. Αυτή η τιμωρία μπορεί να μειώσει την κατανομή πόρων σε χρήστες με σταθερά καλές συνθήκες καναλιού, ευνοώντας αυτούς με κακές. Αυτό έχει ως αποτελέσματα την σημαντική μείωση του συνολικού throughput του συστήματος. Για αυτό και στο σενάριο μας βλέπουμε πως το PF πετυχαίνει χαμηλότερες τιμές και στις δυο μετρικές από ότι το RR. Σε επόμενα σενάρια θα δούμε ότι θα πετύχει και καλύτερο fairness από το RR σε διαφορετικές συνθήκες.

Παρατηρούμε επίσης ότι έχει χαμηλότερο throughput από το Max C/I. Το Max C/I μεγιστοποιεί το throughput του συστήματος, καθώς κατανέμει πόρους σε UEs που διασφαλίζουν την υψηλότερη spectral efficiency. Στα UEs δηλαδή που μπορούν να μεταδίδουν περισσότερα δεδομένα ανά μονάδα χρόνου. Στο σενάριο μας, πράγματι έχει πολύ καλό throughput, όμως έχει χαμηλότερη τιμή fairness από όλους τους scheduler. Αυτό συμβαίνει επειδή οδηγεί πιθανώς τα UEs με κακές συνθήκες καναλιού σε starvation.

Ισορροπία φέρνει ο Exponential Proportional Fair. Για β=2 το βάρος δικαιοσύνης κυριαρχεί και ο scheduler δίνει το μεγαλύτερο fairness από όλους, αλλά υποβαθμίζεται σημαντικά το συνολικό throughput του συστήματος. Παρακάτω βλέπουμε σε ξεχωριστά διαγράμματα ένα πειράματα σύγκρισης του Proportional Fair με τον Exponential για διαφορετικές τιμές του β. Παρατηρούμε πως, όπως περιμένουμε από την θεωρία, για τιμή β=0.1 έχει μεγαλύτερο throughput και χαμηλότερο fairness από το Proportional Fair. Για β=0.5 εξισορροπεί την αποδοτικότητα του throughput και το fairness.



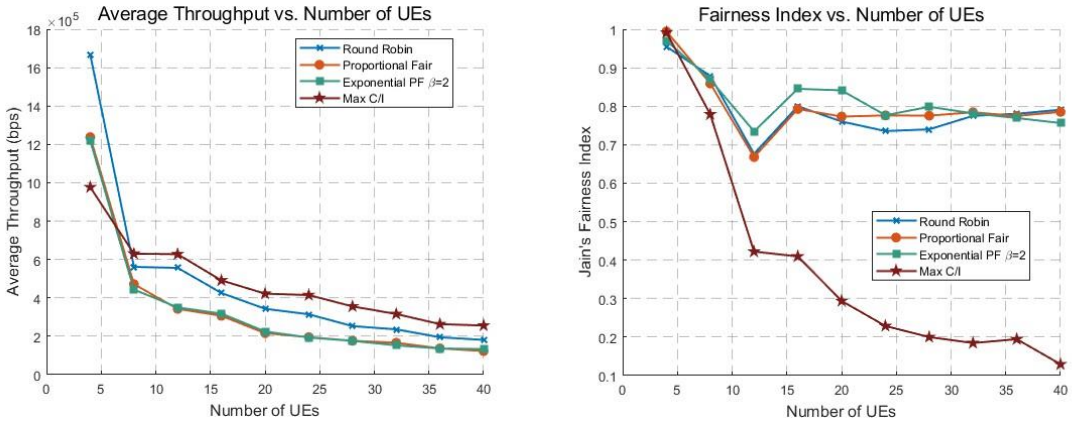
b = 0.1



b = 0.5

Τώρα δοκιμάζουμε με διαθέσιμα 5 και 15 RBGs. Παρατηρούμε πως στο πείραμα με τους λιγότερους διαθέσιμους πόρους έχουμε αποτελέσματα πολύ πιο κοντά στην θεωρία. Ο PF και ο ειδικότερα ο Exponential PF ξεπερνούν και οι δυο τις τιμές fairness του RR, διαχειρίζονται αποδικότερα τους περιορισμένους πόρους. Όταν έχουμε 15 RBGs ο Max C/I και ειδικά ο RR δείχνουν να τους εκμεταλλεύονται με υψηλές τιμές throughput παρά την έλλειψη fairness.

Scheduler	RBGs	Average Throughput	Fairness Index
Round Robin	5	4.7339e+05	0.7892
Proportional Fair	5	3.3733e+05	0.7983
Exponential PF, β=2	5	3.3477e+05	0.8143
Max C/I	5	4.7532e+05	0.3834
Round Robin	15	9.4985e+05	0.6562
Proportional Fair	15	6.3468e+05	0.5109
Exponential PF, β=2	15	6.3903e+05	0.6087
Max C/I	15	5.8911e+05	0.5080



RBGs = 5

FTP Traffic

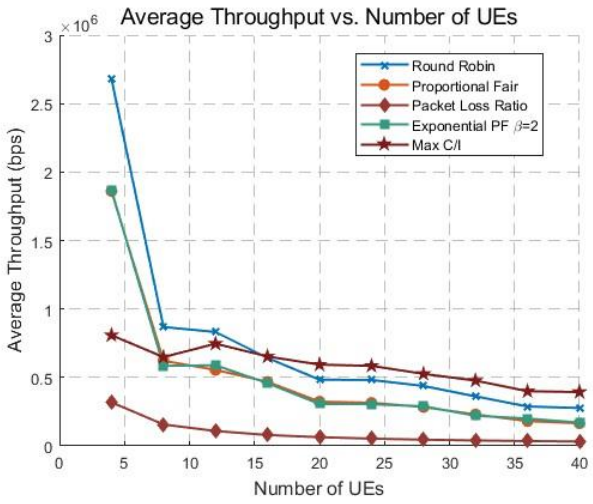
Το FTP (File Transfer Protocol) traffic μοντέλο αναπαριστά μεταφορές αρχείων με μεγάλες περιόδους μετάδοσης πακέτων που ακολουθούνται από περιόδους αδράνειας. Στέλνει αρχεία που αποτελούνται από πακέτα. Το μέγεθος των αρχείων ακολουθεί Pareto distribution. Αυτό

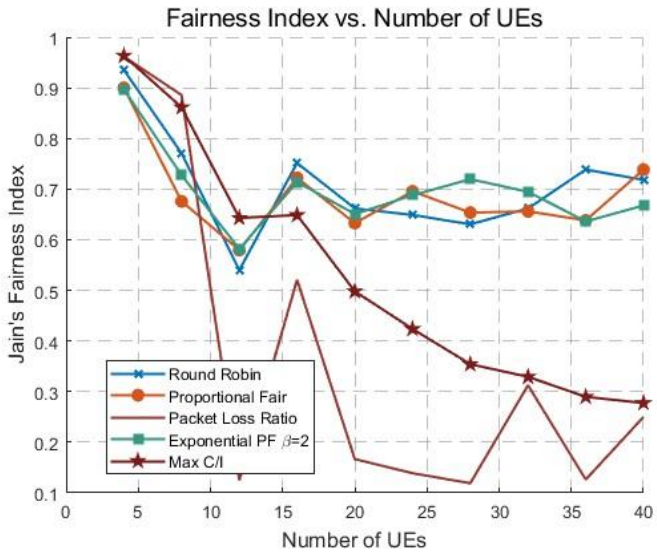
αντικατοπτρίζει την κίνηση στον πραγματικό κόσμο, όπου κυριαρχούν τα μικρά αρχεία, αλλά περιστασιακά μεγάλα αρχεία δημιουργούν αιχμές στο load. Μετά τη μετάδοση όλων των πακέτων για ένα αρχείο, υπάρχει μια καθυστέρηση πριν ξεκινήσει η επόμενη μεταφορά. Ο χρόνος μεταξύ αρχείων ακολουθεί εκθετική κατανομή, για να αντιπροσωπεύει τυχαίες καθυστερήσεις μεταξύ των μεταφορών αρχείων. Τα μεγέθη των πακέτων ορίζεται από εμάς.

Κάνουμε ένα πείραμα που δίνουμε σταθερά 16 packets σε όλα τα UEs.

Scheduler	Iterations	Average Throughput	Fairness Index
Proportional Fair	10	5.0022e+05	0.6892
Round Robin	10	7.3516e+05	0.7057
Exponential PF, β=2	10	4.9467e+05	0.7026
Exponential PF, β=0.5	10	4.9920e+05	0.6098
Exponential PF, β=0.1	10	5.0459e+05	0.6977
Max C/I	10	5.8298e+05	0.5288
Packet Loss Ratio	10	1.3226e+05	0.4748

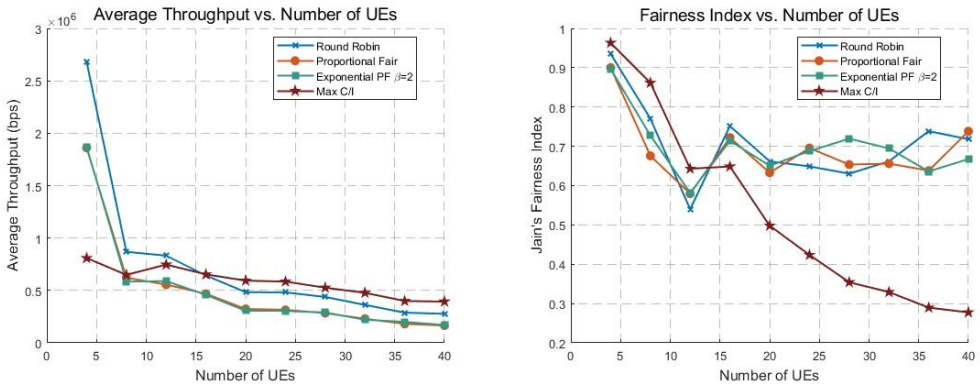
Τα αποτελέσματα είναι πολύ κοντά σε αυτά που είχαμε για Full Buffer Traffic, άρα δε θα επεκταθούμε σε αυτό το κομμάτι. Για το Packet Loss Ratio scheduler βλέπουμε ότι λόγω της προτεραιότητας που δίνει στους UEs με μεγάλο packet loss μένει πίσω σε μεγάλο βαθμό συγκριτικά με τους άλλους schedulers. Μια λύση για την βελτιστοποίηση του θα ήταν να χρησιμοποιήσουμε πληροφορίες για το ιστορικό throughtput ή το στιγμιαίο data rate στην μετρική του, αλλά εκεί θα πηγαίναμε σε διαφορετική υλοποίηση. Με μοντέλο traffic VoIP (Voice over IP) είχαμε καλύτερα αποτελέσματα συγκριτικά με το Proportional Fair, αλλά ήταν ένα μοντέλο που δεν έτρεχε καλά με τους υπόλοιπους schedulers άρα επιλέξαμε να μην συμπεριλάβουμε πειράματα με αυτό. Παρακάτω φαίνονται τα αποτελέσματα:





Τέλος, κάνουμε ένα πείραμα που δίνουμε διαφορετικό αριθμό packet σε group UEs. Σπάμε τον αριθμό τους σε 4. Στο πρώτο group δίνουμε 4 packets, στο δεύτερο 8, τρίτο 12 και τέταρτο 16. Παρατηρούμε πάλι αποτελέσματα κοντά σε αυτά που περιμένουμε.

Scheduler	Average Throughput	Fairness Index
Proportional Fair	5.0022e+05	0.6892
Round Robin	7.3516e+05	0.7057
Exponential PF, β=2	4.9920e+05	0.6977
Max C/I	5.8298e+05	0.5288



Διαφορετικός Αριθμός Πακέτων

Πηγές

- Alsharif, M. H., & Kim, S. (2022). Simulation of 5G Networks using MATLAB. arXiv preprint arXiv:2204.11369. Retrieved from https://arxiv.org/pdf/2204.11369

- MathWorks. 5G Toolbox Documentation. https://www.mathworks.com/help/5g/
- MathWorks. NR FDD Scheduling Performance Evaluation. https://www.mathworks.com/help/5g/
- Hussain, A., Khan, A., & Khan, S. A. (2019). Performance Analysis of Scheduling Algorithms in LTE Networks. International Journal of Simulation: Systems, Science & Technology. https://ijssst.info/Vol-20/No-2/paper17.pdf
- Farhana A., Kumbesan S., Pantha G. Performance Analysis of PF, M-LWDF and EXP/PF Packet Scheduling Algorithms. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7020879
- N. Sirhan N. , Martinez-Ramon M. LTE CELLULAR NETWORKS PACKET SCHEDULING ALGORITHMS IN DOWNLINK AND UPLINK TRANSMISSION: A SURVEY. https://arxiv.org/pdf/2204.11369