**Задачи по планировщику**

Ресурсная сетка есть, какие пригодятся методы я реализовал, возможно что-то потребуется нового либо я где-то накосячил – согласуй со мной эти моменты, пиши при любых вопросах, я отвечу, как будет время, хоть его и мало. Если к чему-то страшно прикасаться – не прикасайся)). Код для LTE\_GRID лучше изучить внимательно, я где мог коментил, сделал документашку.

Текущие твои наработки по планировщикам никуда не пропадут. Используем. Но нужно отредачить код чтобы функционал соответствовал текущей решетке. А она приближена к реальной, если не учитывать, что мы не опустились до RE, ибо туда оочень страшно лезть и больно то надо пока.

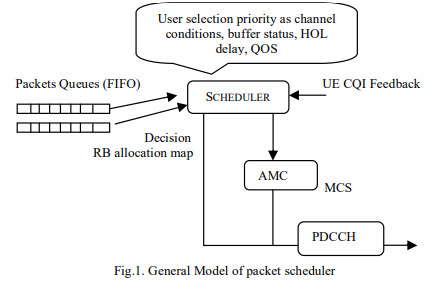
1. По реализации планировщика.

Создал класс SchedulerInterface (скорее будем переименовывать).

Все алгоритмы должны наследоваться от этого класса и реализовывать метод schedule. Функцию для инициализации я уже написал, как и для других блоков.

А вообще, можешь переписать как хочешь, я просто написал, чтобы было понятно, что подтягивать в класс.

Планировщик должен запрашивать у ресурсной сетки список доступных RB через метод **GET\_FREE\_RB\_FOR\_TTI()**, а затем выделять их пользователям через **allocate\_rb()** или **allocate\_rb\_group()**. Заметь, на выходе из планировщика у нас decision\_map (поищи, метод для неё я написал в коде, можно юзать его). А вообще, надо выпилить его нахрен из файла LTE\_GRID. Потому что если сломаешь LTE\_GRID собирая планировщик – плакать будем очень долго. Нейронки нам тут не помогут восстановить потери, либо накосячат. Остальные методы допишем потихоньку. Напоминаю примерную схему планировщика.



2. По юзерам. Я не стал создавать класс для пользователей, чтобы оставить тебе возможность интегрировать свой кусок кода, а не делать все с нулины. Но помни, твой юзер должен обладать след. атрибутами:

* id: уникальный идентификатор (в коде UE\_ID)
* velocity: скорость движения в м/с
* channel\_quality: качество канала связи (можно выразить через SINR или CQI)
* buffer status – чего там у него с буфером
* mcs – текущий мцс
* далее, что тебе будет необходимо

3. Модели движения абонентов. Лучше сделать это отдельным модулем (и соотв. классом). Модели нужно дать возможность менять. Реализовать модели можно через методы. Пример моделей ты знаешь. Random Walk и так далее. Гаусса-Маркова кстати одна из самых реалистичных и тяжелых. Но можно и модель передвижения в городе.

Поскольку время в симуляции у нас измеряется в миллисекундах (зависит от количества кадров) а заметное перемещение пользователя происходит гораздо реже (а вообще по разному, если это спутник или поезд то тут конечно мда), предлагаю обновлять положение юзера не каждый TTI, а например, каждые 100-500 мс, и так далее. Вообще, надо погуглить, как часто базуха запрашивает статус канала от юзера. Скорее всего часто по нескольку раз в кадре. Но у нас таких вычислительных ресурсов нет. Ничего страшного, что время опроса качества канала будет редким, это регулируемый параметр базух. А в идеале! Сделать метод, который позволил бы регулировать этот параметр. Ну и всегда весело поискать предел, на котором у нас просто все крашнется к чертям собачим.

Итак по моделям надеюсь понятно? Реализовать разные модели, модели можно менять. Не забудь мат. Обоснование моделей

4. Модели канала. Окамура-Хата (устарела), COST321-Hata и так далее. Но важно понимать, что так-то они имеют ограничения по частотам и другим параметрам. Вообще в документашке 3GPP уже давно предложены модели, а может где-то и код реализован. Тут от нас не требуется париться и писать все. Есть возможность катать – катаем.

Как подружить модель канала и перемещение абонов – думаю ты уже знаешь, но я не смотрел внимательно твой код в этой реализации. Пока для меня это какие-то чудеса. Хотя в целом предполагаю, что ты меняешь дистанцию в моделях, в зависимости от передвижения.

Современные модели канала статистические. Т.е. инженеры также как и Окамура ходили в различной местности и замеряли уровни сигнала, а затем в математической формуле подгоняли коэффициенты чтоб цифры бились. Иными словами, четкой математической закономерности там нет, только статистика.

Влияние Доплера и многолучевости? Ну…да, можно как методы сделать…потом. Когда-нибудь. Йес.

Модели опять же вызывать будем в тестовой среде и там же их менять.

5. Модели трафика – это самое страшное. Пока сюда не лезем. Там реально очень страшно. Но есть подозрения, что есть какая-то математическая принятая модель. Возможно, нам подскажут Саша с Мишей что они нарыли. В матлабе то уже все есть.

Рано или поздно мы дойдем до QoS. И его придется учитывать во всех модулях…

6. Запросы на статус буфера, CQI, и так далее – их нет. Надо реализовывать в соответствующих классах эти методы.

7. Модуль адаптивной модуляции. АМС. Отдельная тема, как он работает. Это надо изучить, я еще туда не лез, но возможно функционал его достаточно прост, по качеству канала смотрит какие варианты модуляции дать юзеру. Полностью мы его моделировать не будем. Только имитировать ( а может и даже пародировать ахаха). Потому что, понятное дело, если мы меняем схему модуляции, об этом должен узнать юзер (думаю вы на sdr это проходите, понимаешь механизм). Значит должна быть сигнализация. Нет времени на иммитацию сигнализации. Делаем все по делу.