**Документация по модулю LTE\_GRID**

**1. Немного теории**

Модуль представляет собой программную реализацию частотно-временной ресурсной сетки сети LTE на языке Python. Модель позволяет симулировать распределение ресурсных блоков между пользователями и может использоваться для тестирования различных алгоритмов планирования ресурсов (MAC scheduler).

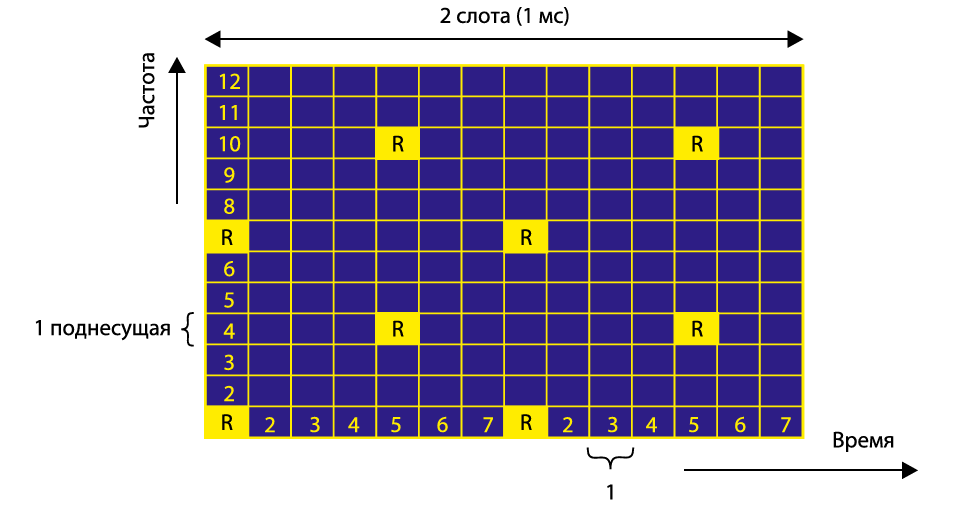
В LTE радиоресурсы организованы в структурированную сетку как по частоте и по времени. Понимание этой структуры критически важно для эффективного распределения ресурсов между пользователями и достижения максимальной спектральной эффективности.

**2. Структура ресурсной сетки LTE**

Ресурсная сетка LTE имеет иерархическую структуру с несколькими уровнями организации:

* **Ресурсный блок (Resource Block, RB)** - минимальная единица распределения ресурсов, состоит из 12 поднесущих в частотной области и 1 слота во временной.
* **Слот (Slot)** - временной интервал длительностью 0.5 мс, содержит набор ресурсных блоков по частоте.
* **Подкадр (Subframe)** - временной интервал длительностью 1 мс, состоит из двух последовательных слотов. Также известен как TTI (Transmission Time Interval).
* **Кадр (Frame)** - временной интервал длительностью 10 мс, содержит 10 подкадров.

В зависимости от выделенной полосы частот (от 1.4 МГц до 20 МГц), количество ресурсных блоков по частоте может варьироваться от 6 до 100 согласно спецификации 3GPP (см. далее).



**2.1 Ресурсный блок (RB)**

В LTE ресурсный блок состоит из 12 поднесущих (subcarriers) в частотной области, с шагом 15 кГц между поднесущими, и одного слота (0.5 мс) во временной области. Таким образом, каждый RB занимает полосу 180 кГц (12 × 15 кГц) и имеет длительность 0.5 мс.

Ресурсный блок содержит ресурсные элементы (Resource Element, RE), каждый из которых представляет одну поднесущую на протяжении одного символа OFDM. Количество символов OFDM в слоте зависит от типа циклического префикса (CP):

* Нормальный CP: 7 символов OFDM в слоте
* Расширенный CP: 6 символов OFDM в слоте  
  (в коде представил оба варианта CP и сетка будет перенастраиваться)

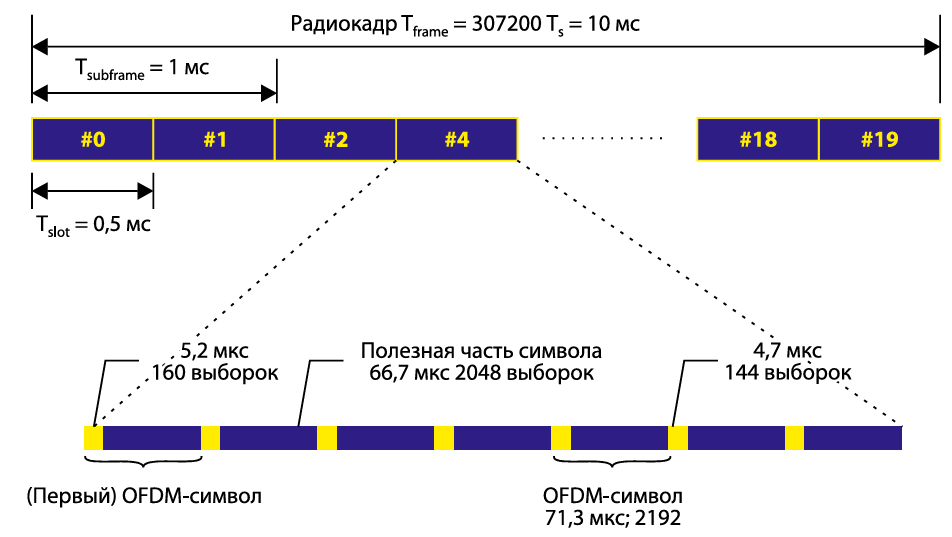
**2.2 Полоса частот и количество RB**

Согласно спецификации 3GPP, в LTE определены следующие конфигурации полосы частот и соответствующее им количество ресурсных блоков:



**Временная структура**

* Слот (Slot): 0.5 мс
* Подкадр (Subframe): 1 мс, состоит из 2 слотов
* Кадр (Frame): 10 мс, состоит из 10 подкадров



**2.3 Дополнительные ресурсы**

* 3GPP TS 36.211: "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation"
* 3GPP TS 36.213: "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures"
* 3GPP TS 36.300: "Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description"

Доки содержат полную спецификацию физического уровня LTE, включая детальное описание структуры ресурсной сетки и правил распределения ресурсов. Пока реализуем LTE. Если мы начнем мешать всю документашку еще и с 5G, учитывая там всякие нумерологии гибкие фреймы и так далее – мы точно крышей поедем.

**3. Описание классов модели**

**3.1 Класс RES\_BLCK**

Базовый класс, представляющий ресурсный блок - минимальную единицу распределения ресурсов в LTE.

**class RES\_BLCK:** def \_\_init\_\_(self, id: str, time\_idx: int, freq\_idx: int):  
 """  
 Инициализация ресурсного блока.  
   
 Args:  
 id: Уникальный идентификатор блока  
 time\_idx: Временной индекс (в рамках TTI)  
 freq\_idx: Частотный индекс  
 """

**Основные методы:**

* allocate(UE\_ID) - назначает ресурсный блок пользователю с указанным идентификатором
* CHCK\_RB() - проверяет, свободен ли ресурсный блок

**Атрибуты:**

* id - уникальный идентификатор блока
* time\_idx - временной индекс блока
* freq\_idx - частотный индекс блока
* UE\_ID - ID пользователя, которому назначен блок (None если блок свободен)
* status - статус блока ("free" или "allocated")

**3.2 Класс Slot**

Класс, представляющий слот в структуре LTE. Слот содержит набор ресурсных блоков по частоте.

**class Slot:**  
 def \_\_init\_\_(self, slot\_id: int, num\_rb: int):  
 """  
 Инициализация слота.  
   
 Args:  
 slot\_id: Идентификатор слота  
 num\_rb: Количество ресурсных блоков по частоте  
 """

**Основные методы:**

* GET\_RES\_BLCK(rb\_fid) - получить ресурсный блок по частотному индексу
* GET\_ALL\_RES\_BLCK() - получить все ресурсные блоки в слоте
* GET\_FREE\_RES\_BLCK() - получить все свободные ресурсные блоки в слоте

**Атрибуты:**

* id - идентификатор слота
* resource\_blocks - словарь ресурсных блоков в слоте

**3.3 Класс Subframe**

Класс, представляющий подкадр (TTI) в структуре LTE. Подкадр состоит из 2 слотов.

**class Subframe:**  
 def \_\_init\_\_(self, subframe\_id: int, num\_rb: int):  
 """  
 Инициализация подкадра.  
   
 Args:  
 subframe\_id: Идентификатор подкадра  
 num\_rb: Количество ресурсных блоков по частоте  
 """

**Основные методы:**

* GET\_SLOT(slot\_idx) - получить слот по индексу
* GET\_ALL\_RES\_BLCK() - получить все ресурсные блоки в подкадре
* GET\_FREE\_RES\_BLCK() - получить все свободные ресурсные блоки в подкадре

**Атрибуты:**

* id - идентификатор подкадра
* slots - список слотов в подкадре (обычно 2)

**3.4 Класс Frame**

Класс, представляющий кадр в структуре LTE. Кадр состоит из 10 подкадров.

**class Frame:**  
 def \_\_init\_\_(self, frame\_id: int, num\_rb: int):  
 """  
 Инициализация кадра.  
   
 Args:  
 frame\_id: Идентификатор кадра  
 num\_rb: Количество ресурсных блоков по частоте  
 """

**Основные методы:**

* GET\_SUBFRAME(subframe\_idx) - получить подкадр по индексу
* GET\_ALL\_RES\_BLCK() - получить все ресурсные блоки в кадре
* GET\_FREE\_RES\_BLCK() - получить все свободные ресурсные блоки в кадре

**Атрибуты:**

* id - идентификатор кадра
* subframes - список подкадров в кадре (обычно 10)

**3.5 Класс RES\_GRID\_LTE**

Основной класс для моделирования ресурсной сетки LTE. Управляет кадрами, подкадрами, слотами и ресурсными блоками.

**class RES\_GRID\_LTE:**  
 def \_\_init\_\_(self, bandwidth: float = 10, num\_frames: int = 10, cp\_type: str = "normal"):  
 """  
 Инициализация ресурсной сетки LTE.  
   
 Args:  
 bandwidth: Полоса частот в МГц (1.4, 3, 5, 10, 15, 20)  
 num\_frames: Количество кадров для симуляции  
 cp\_type: Тип циклического префикса ("normal" или "extended")  
 """

**Основные методы:**

* GET\_RB(tti, freq\_idx) - получить ресурсный блок по TTI и частотному индексу
* allocate\_rb(tti, freq\_idx, UE\_ID) - назначить ресурсный блок пользователю
* allocate\_rb\_group(tti, freq\_indices, UE\_ID) - назначить группу ресурсных блоков пользователю
* GET\_FRAME(frame\_idx) - получить кадр по индексу
* GET\_SUBFRAME(tti) - получить подкадр по TTI
* GET\_FREE\_RB\_FOR\_TTI(tti) - получить список свободных RB для заданного TTI
* GET\_TTI\_STATUS(tti) - получить статус распределения ресурсных блоков для заданного TTI
* NEXT\_TTI() - переход к следующему TTI
* GET\_GRID\_STATUS() - получить текущий статус всей ресурсной сетки
* GET\_STATS() - получить статистику использования ресурсов
* visualize\_grid(tti\_start, tti\_end, show\_UE\_IDs, save\_path) - визуализация ресурсной сетки
* visualize\_statistics(save\_path) - визуализация статистики использования ресурсов

**Ключевые атрибуты:**

* bandwidth - полоса частот в МГц
* num\_rb - количество ресурсных блоков по частоте
* num\_frames - количество кадров для симуляции
* cp\_type - тип циклического префикса
* total\_tti - общее количество TTI (подкадров)
* frames - список кадров
* rb\_map - словарь для быстрого доступа к RB по TTI и частотному индексу
* current\_tti - текущий TTI
* stats - статистика использования ресурсов

**Константы:**

* BANDWIDTH\_TO\_RB - словарь соответствия полосы частот и количества RB согласно стандарту LTE

**4. Примеры использования**

**4.1 Создание ресурсной сетки**

# Создание ресурсной сетки с полосой 10 МГц (50 RB) и 2 кадрами (20 TTI)  
lte\_grid = RES\_GRID\_LTE(bandwidth=10, num\_frames=2)  
  
print(f"Создана сетка с {lte\_grid.num\_rb} RB в каждом TTI, всего {lte\_grid.total\_tti} TTI")

**4.2 Назначение ресурсных блоков**

# Назначение отдельного ресурсного блока  
lte\_grid.allocate\_rb(tti=0, freq\_idx=5, UE\_ID=1)  
  
# Назначение группы ресурсных блоков  
lte\_grid.allocate\_rb\_group(tti=1, freq\_indices=[10, 11, 12, 13], UE\_ID=2)

**4.3 Получение свободных ресурсных блоков**

# Получение списка свободных RB для TTI 0  
free\_rbs = lte\_grid.GET\_FREE\_RB\_FOR\_TTI(0)  
print(f"Доступно {len(free\_rbs)} свободных RB в TTI 0")

**4.4 Визуализация ресурсной сетки**

# Визуализация первых 10 TTI  
lte\_grid.visualize\_grid(tti\_start=0, tti\_end=10)  
  
# Визуализация с сохранением изображения  
lte\_grid.visualize\_grid(tti\_start=0, tti\_end=20, save\_path="lte\_grid.png")

**4.5 Анализ статистики**

# Получение статистики использования ресурсов  
stats = lte\_grid.GET\_STATS()  
print(f"Всего ресурсных блоков: {stats['total\_rbs']}")  
print(f"Назначено ресурсных блоков: {stats['allocated\_rbs']}")  
print(f"Процент использования: {stats['utilization\_percent']:.2f}%")  
  
# Визуализация статистики  
lte\_grid.visualize\_statistics()