|  |
| --- |
| Jpretty核心项目攻坚组 |
| **LTE上下行资源分配** |
| Resource Allocation For LTE`s Uplink And Downlink |

|  |
| --- |
| Jpretty  2016/7/24 |

## LTE下行资源分配

所谓下行资源分配，表达的是针对下行的频域时域网格如何划分给每一个用户，针对每一个下行用户(UE)，其下行传输是由基站(eNodeB)控制的。针对每一个TTI内的资源分配情况，UE会在PDCCH中搜索其对应的DCI，DCI中有字段通知UE该TTI内的下行数据在哪些RB上进行传输。以下介绍主要参考协议TS36.211-7.1.6。

UE通过对应的PDCCH中的DCI消息的检测得到对应的eNodeB分配给UE的资源分配方案(这些需要解释的是UE如果才能从众多的DCI消息中找到自己的PDSCH对应的DCI消息，这个是通过DCI消息不同的加扰序列确认的)。在每一个DCI消息中用于资源分配的字段包括两个部分：Head域和实际的分配方案信息。

其中Head域的作用协议指出，DCI 1/2/2A/2B/2C中用于Type0和Type1的实际资源分配方案信息是完全一致的，所以通过1bit的Head域来指示UE使用以上两种类型的哪一种。另外一半来说上述的DCI 1/2/2A/2B/2C采用的是Type0或者Type1中的一种，而剩余类型的DCI 1A/1B/1C/1D常用资源分配类型Type2，且Type2类型的DCI信息中没有Head域。

下面针对不同的资源分配类型进行详述，同时需要给出的是以下的分配方案均是建立在VRB的基础上的，后续会针对VRB到PRB的映射(分布式或者集中式等等做介绍)。在此之前先大致给一个对于三种下行资源分配类型的简单介绍：

* 1. 下行资源分配Type0是最简单的分配方案，其中的分配粒度是RBG(RB组)
  2. 下行资源分配Type1在Type0的基础上，以RBG子集中的RB为分配粒度的方案
  3. 下行资源分配Type2是分配一段连续的VRB给UE的方案，通过RIV值指示

### 下行资源分配Type0

如上所述，下行资源分配Type0是最简单的资源分配方案，其中将若干个连续的虚拟资源块(VRB)组成一个RBG，这样整个下行带宽就被分成若干个RBG集合，然后DCI消息中通过一个比特序列(bitmap)来表示对应哪些RBG用于下行数据传输。

其中RBG的大小，即一个RBG中包含的RB的个数，是由具体的该小区下行带宽决定的，协议TS 36-213中Table 7.1.6.1-1给出了解释。其中第一列表示对应的下行带宽，第二列表示该下行带宽下RBG占据的RB个数。其中需要指出的是，当整数个RBG无法全部占满所有的下行带宽上的RB的时候，最后一个RBG中包含的RB数目要比前面的RBG中包含的RB数少(显而易见)。

当RBG划分好后(假设一共有个RBG)，那么DCI消息中会就会有大小为的一个Bitmap用于指示对应的哪些RBG用于下行传输，Bitmap中比特的映射准则是MSB映射到RB标号较低的RBG，LSB映射到RB标号较高的RBG，如果对应的比特值为1，那么表示该RBG针对这个UE被用作下行数据传输，反之如果该比特为0，就表示不是用来为该UE进行传输的。

|  |  |
| --- | --- |
| System Bandwidth | RBG Size |
| C:\Users\ADMINI~1\AppData\Local\Temp\ksohtml\wps197.tmp.png | (*P*) |
| ≤10 | 1 |
| 11 – 26 | 2 |
| 27 – 63 | 3 |
| 64 – 110 | 4 |

虽然Type0方案比较简单，还是通过举个例子来说明一下上述的阐述，假设该小区的下行带宽为5MHz，也就是说下行一共分配了25个RB用来进行下行传输，查阅上表中的第一列当下行分配的RB数处于11至26之间的时候，对应的RBG中的RB个数为2，所以分配好的RBG为个，其中第0至第11个RBG的大小均为2，最后一个RBG中仅有1个RB。当DCI消息中用于指示哪些RBG用来传输的字段为0010111101010时，下行RBG的分配情况如下图所示。其中用于下行传输的RB用红色底色表示。



从上述表述可以看到，Type0的资源分配类型仅能用于RBG粒度的分配，这样分配虽然简单，但是存在问题，例如在带宽比较大的时候分配粒度太大(最大支持4个RB)，分配方式不高效，所以接下来介绍下行资源分配Type1。

### 下行资源分配Type1

下行资源分配Type1其DCI中用于表示RB分配信息的长度和Type0的保持不变，同样为长度。但是其分配方式和Type0完全不同，Type1将上述的大小为P的个RBG分成P个子集，然后通过上述比特来指示在这些子集中选择哪一个子集中的哪些RB来为下行传输提供资源。其中子集的划分在协议中表示的是第p个子集从RBG标号为p开始的每P个RBG作为一个子集的，

这样从文字上来说可能比较难以理解，可以按照下方的示例图进行解释。其中划分好的RBG按照标号按先列后行的顺序存入P列的矩阵中，然后每一列表示一个RBG子集，其中需要注意的是，划分好的RBG的个数可能不是P的倍数，所以该矩阵可能无法完全放置完全，所以位于后方的RBG子集可能存在其中包含的RBG个数要小于位于前方的RBG子集中的RBG个数的情况。



在DCI中用于确认下行资源分配方案的信息中，首先应该有个比特用于指示到底选择一共P个RBG子集中的哪一个用来分配。

其次第二个部分用于指示在对应的RBG子集中是否存在偏置，偏置的这个问题需要额外阐述一下，可以想象的是，如果没有这个偏置的话。由于有部分比特已经用来表示选择哪个子集用于发送了，剩余的比特可能存在表示不完该子集中的所有RB的情况，那么就会存在矩阵上方的RB始终能够被分配，而处于矩阵下方的RB无法被分配的情况，这样显然是不高效的，所以使用1比特的偏置标志位来表示在RBG字节内部是否进行偏置。其中比特1表示在RBG字节内部要进行偏置，比特0表示不进行偏置。

第三个部分也就是剩余的部分用来表示在该RBG子集中哪些RB用来进行下行传输，这个稍微复杂一点，首先除开第一和第二个部分的比特剩余的比特长度可以表示为：



如果第二个部分表示的偏置为0的话就代表的对应选择的RBG子集内部不做偏置，即上述的个比特表示从RBG子集内部第一个RB开始，持续个RB的下行分配情况，其中对应的比特表示1的话就说明这个RB分配给这个UE，否则表示这个RB没有分配给这个UE。

如果第二个部分表示的偏置为1的话就代表对应选择的RBG子集内部要做偏置，即上述的个比特表示从RBG子集内部的最后一个RB开始，倒序持续个RB的下行分配情况，其中对应的比特表示1的话就说明这个RB分配给这个UE，否则表示这个RB没有分配给这个UE。

协议中在这个地方的阐述比较复杂，其中涉及到每个RBG子集中占据的RB的个数计算如下表所示：



其中可以看到的是，每一个分支的判断条件的位置，其实判断的就是对应的子集标号，也就是p的位置，其中区分的标准在于每一个RBG子集位于矩阵最后一行的RB的个数。协议为了方便编程很多算法类型的问题都采用了公式化的方法去解释，在这里还是举个例子来说明，如果该小区的下行带宽为10M，那么对应的RB个数为50个，查表可知每个RBG的大小为3，所以按照上述推导，50个RB被划分成17个RBG(16个RBG大小为3，最后一个RBG的大小为2)，子集划分之后的RBG矩阵形式如下所示。



现在考虑上述表达式中的第一个分支，分支表达式为：



所以p等于0的时候表示的是第一列所表示的RBG子集，所以该子集中的RBG个数可以分为两部分，第一部分是0/3/6/9/12中的RB个数，其中每个RBG中的RB个数保持一致，为：



第二部分是最后一行的RBG15，该RBG满足第一个分支条件，所以其中的RB个数也是3，所以RBG子集0中的RB个数为18个。同理，当p=1的时候满足第二个分支条件，此时相对于前一列RBG子集0来说，区别在于最后一行的RBG16在进行RBG划分时本身处于不满的状态，该RBG16中的RB个数为2，所以需要额外处理后RBG子集1中的RB总个数为17个。最后一个分支条件对应RBG子集2，也就是矩阵的最后一列，该列由于没有最后一行，所以仅有若干个RB个数相同的RBG子集构成，对应一共15个RB。

再然后是RBG子集内部的RB调度情况，前面提到过有一个比特用于偏置表示，如果该比特为0，那么就从第一行对应的RBG开始分配，反之则从最后一行的RBG开始分配，例如上面这个例子，如果2比特的RBG子集标志位确定使用RBG子集1，且偏置标志位为0的话，那么DCI中的长度为17比特的资源分配Bitmap除开前面两个部分共3比特后剩余的14比特用于指示RBG1/4/7/10/13/16中从RBG1中的第一个RB开始分配，持续14个RB的分配情况，正如前面所述，RBG子集1一共17个RB，这样分配后面3个RB始终没法分配。所以如果存在偏置的话，这14个比特表示的就是RBG4/7/10/13/16这5个RBG中共14个RB的资源分配方案了。

至此完成了下行资源分配Type1的介绍，可以看到的是，和Type0不同的是虽然两者在资源分配方案信息位虽然保持一致，但是Type1可以使得分配粒度降低到每个RB，但是由于初始选择是基于RBG子集得到的，所以在实际使用中虽然粒度变小，但是个人觉得灵活性较Type0会有所下降。

### 下行资源分配Type2

最后需要阐述的是下行资源分配Type2，Type2相对于上述的Type0和Type1有较大区别，且Type2也用作上行资源分配中，所以下面详细阐述。下行资源分配Type2中通过DCI中的资源分配信息指示为UE分配了连续的VRB用来针对UE进行下行传输。协议中有提到分布式VRB和集中式VRB的区别，这里其实我们并不需要关注，分布式或集中式VRB是指的是VRB至PRB的映射过程，对于资源分配来说都是针对VRB来做的，所以此处需要重视的是Type2中的VRB分配均是连续的。同时，采用的分布式还是集中式的VRB，针对DCI 1A/1B/1D来说存在一个比特的标志位来确认，对于DCI 1C来说常常采用分布式的VRB映射方法，上述DCI 1A/1B/1C/1D是常用的Type2的DCI格式。

针对集中式的VRB分配可分配的VRB从1个到下行带宽支持的最大VRB数之间选择，针对于配置分布式VRB分配方案的DCI 1A来说，当SCI通过P-RNTI/RA-RNTI/SI-RNTI加扰时，可以分配的VRB数目从1个到个(这个参数以后的文档会详述)。针对使用C-RNTI加扰的DCI 1B/1D，以及使用C-RNTI/SPS-C-RNTI/T-C-RNTI加扰的DCI 1A来说，如果下行RB数小于50，那么可以分配的VRB数和上述的区间一致；而当下行RB数大于等于50的时候，可分配的VRB数从1最大可至16。针对DCI 1C来说常常仅能使用Type2类型的资源分配方式，其分配是按照步长进行的(步长大小按照带宽大小可以取2或者4)，所以DCI 1C的VRB分配数可以为2或者4的整数倍。

针对DCI 1A/1B/1D来说，下行资源分配类型Type2中的资源分配信息可以通过一个整数RIV来表示对应的RB分配的起点和持续的连续VRB的个数，其中通过这两个参数可以知道具体分配的哪些RB，RIV的取值如下所示：

* 当时，
* 当时，

解决了如何从RB分配起点和连续RB的持续个数得到RIV的问题后，当UE从DCI中获得对应的RIV后，怎么从RIV中得到具体的资源分配方案，也就是RB起点和RB的持续个数。可以看到的是协议设计都是很巧妙的，首先进行如下计算：



如果RIV是通过第一个分支得到的话，那么：



所以此时可以得到：



同理如果RIV是通过第二个分支得到的话，那么：



所以可以得到：



针对DCI 1C来说，其基本的做法和DCI 1A/1B/1D的相同，只不过针对上述计算中的参数均由其对应的步进参数替代。从上述的表述中可以看到，Type2类型的资源分配方案，相比较Type1和Type0来说更为固定，其分配的RB数目仅仅是连续的若干VRB的集合。

## LTE上行资源分配

上行资源分配的方案相对于下行来说更为固定，这是由于在上行资源分配时，只能分配连续的VRB给UE进行上行传输，这是由于如果分配的是不连续的VRB的话会破坏上行传输时的单载波特性(为什么分配不连续的VRB就会破坏上行单载波特性，之后的文档会详述)。

上行资源分配方案分为Type0和Type1，其中Type1是在对应的R10版本中才提出的。在进行对应的具体分析之前同下行时相同给出大致的总结。

1. 上行资源分配方案Type0和下行资源分配方案Type2保持一致，仅修改对应的参数从下行参数至上行参数
2. 上行资源分配方案Type1可以看做上行资源分配方案Type0的交叉换序形式，即分配了两个类似Type0的连续VRB集合，然后时分选择两个连续VRB集合进行上行数据发送，从而获得频率分集收益。

### 上行资源分配Type0

如前所述，上行资源分配Type0和下行资源分配Type2保持一致，即使用RIV值获取对应的RB开始位置和持续的RB个数，仅仅将对应的例如下行带宽参数变为对应的上行带宽参数等，此处不再详述(以后需要补充的时候会更新版本)。

### 上行资源分配Type1

上行资源分配Type1定义了两段连续的VRB范围用于频率分集的时分资源使用，所以个人认为其实Type1可以称为增强型Type0。但是UE通过DCI中解到的RIV消息来获得两段连续VRB的范围的话RIV值的比特数可能就不够了，所以此时需要改变其对应的表达方式。这里协议中采用的组合数组来表示的，但是在R10版本的协议中并没有针对上行资源分配做特殊的介绍，组合数组是通过CQI反馈一节中提到(如果以后看到了会做相对应版本的刷新)。

要完成两个Type0的DCI消息通报UE，最简单的方式就是通过4个值：

* 第一个Type0的起点RB号
* 第一个Type0的终点RB号
* 第二个Type0的起点RB号
* 第二个Type0的终点RB号

那么用一个组合数r便可以唯一得表示这四个数，其组合数形式为：



其中每个方框代表的是组合数，数学理论表示这样的组合数序列是唯一的，所以在DCI中只要进行r的传输，UE就可以在解到r以后获得正确的两个Type0的位置信息。例如上述的四个位置的值分别为3/10/12/15，那么对应的组合数为：



当UE通过接收DCI时获得1633这个数以后，可以通过贪婪算法获得对应的四个数的组合，具体的步骤是：

* 首先选择使得小于等于1633最大的数作为的值，即，所以取值为15，除开第一个组合数后剩下的还有1633-1365=268
* 然后选择使得小于等于268的最大数作为的值，即，所以取值为12，除开第二个组合数后剩下的还有268-220=48
* 然后选择使得小于等于48的最大数作为的值，即，所以取值为10，除开第三个组合数后剩下的还有48-45=3
* 最后剩余的数作为的值，所以自此就得到了两个Type0的所有资源分配方案参数