## 2022/11/23 测试日志

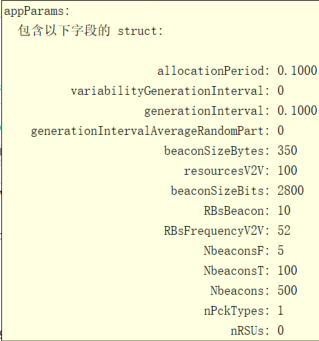
**测试目的：**测试CV2XsensingProcedure.m和BRreassignment3GPPautonomous.m的运行流程，从中验证自己写的伪代码是否正确。

### 测试1：

在CV2XsensingProcedure.m的第一行有效行设置断点观察中间变量的变化情况。

currentT = mod(timeManagement.elapsedTime\_TTIs-1,appParams.NbeaconsT)+1;

首先观察appParams结构体里相关的变量具体值。



由上可知：

|  |  |
| --- | --- |
| **属性** | **值** |
| appParams.NbeaconsF | 5 |
| appParams.NbeaconsT | 100 |
| appParams.Nbeacons | 500 |
| appParams.RBsFrequencyV2V | 52 |

而关于appParams.NbeaconsF和appParams.NbeaconsT的计算是在MatFilesUtility5G\calculateNB\_5G.m中完成的。而对于NR支持的子信道尺寸为[10, 15, 20, 25, 50, 75, 100]。这次实验选择的是尺寸为10的子信道尺寸。关于NbeaconsF的计算公式为：

**appParams.NbeaconsF**

**= floor(appParams.RBsFrequencyV2V/(phyParams.RBsBeaconSubchannel));**

**=floor(52/10)=5**

上面的变量表示的是在频域上beacons的数量，而phyParams.RBsBeaconSubchannel表示的则是在每个beacon上有多少个子信道。

（关于**phyParams.NsubchannelsBeacon**怎么得1的目前还不知道，等下再看）

可用子信道数随SCS和PRB带宽变化，如果假设10个PRB的子信道尺寸，那么在给定的信道带宽10MHz情况下，SCS=15kHz可以容纳5个子信道。（很好理解，就是52个PRBs，而一个子信道智能容纳10个PRBs，所以至少需要5个子信道）。

而关于appParams.NbeaconsT的计算公式为：

appParams.NbeaconsT = floor(appParams.allocationPeriod/phyParams.Tslot\_NR);

=floor(0.1/0.001)=100

上面的phyParams.Tslot\_NR的计算公式为:

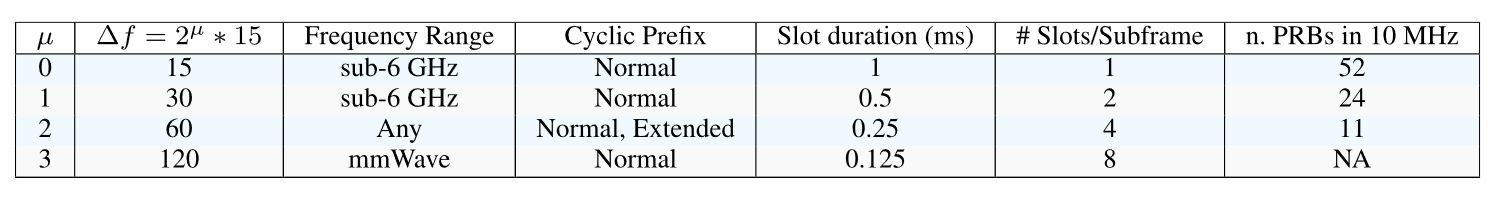
phyParams.Tslot\_NR = 1e-3/(phyParams.SCS\_NR/15);

它表示的是在一个分配周期内有多少个slot，在这种情况下，有100个slot即时域上beacons的数量。那么总的beacons数量即为时域乘上频率的数量即100\*5=500个beacons资源。

因为NR中subframe的时间为0.001，而对于不同SCS的slot结果为：

|  |  |
| --- | --- |
| SCS/kHz | 时间/s |
| 15kHz | 0.001 |
| 30kHz | 0.0005 |
| 60kHz | 0.00025 |

其中RBsFrequencyV2V是由MatFilesUtilityUtility5G\RBtable\_5G.m中查询RBperChannel.txt对应的值得到的，因为SCS=15kHz，而BandWidth=10MHz，所以查询得到为52。



注：上表倒数第二列表示的是每个子帧的slot数，比如第一个，即一个子帧包含1个slot。

CurrentT= mod(timeManagement.elapsedTime\_TTIs-1,appParams.NbeaconsT)+1

= 1

因为一个subframe就是一个slot，所以在CurrentT=1的情况下，我们知道了当前的BRids\_currentSF。

BRids\_currentSF

= ((currentT-1)\*appParams.NbeaconsF+1):(currentT\*appParams.NbeaconsF)

= (0\*5+1):(1\*5)

= [1, 2, 3, 4, 5]

紧接着就需要对感知矩阵进行移位操作，首先来讲述一下感知矩阵的维度组成。对感知矩阵的初始化是在Maininit.m上完成的：

stationManagement.sensingMatrixCV2X

=zeros(ceil(simParams.TsensingPeriod/appParams.allocationPeriod),appParams.Nbeacons,simValues.maxID);

=zeros(ceil(1/0.1),500,2000/1000\*100)=zeros(10,500,200)

然后进行对感知矩阵进行移位，将最后一行的即就是最后一次测量的值放在一位使用：

stationManagement.sensingMatrixCV2X(:,BRids\_currentSF,:)

= circshift(stationManagement.sensingMatrixCV2X(:,BRids\_currentSF,:),1);

这样就能得到第十次的数据即最后测量的。

接下来就是初始化感知功率矩阵：

sensedPowerCurrentSF\_MHz

= zeros(length(BRids\_currentSF),length(stationManagement.activeIDsCV2X));

=zeros(5,200)

它表示的即是每辆车对应每个候选资源的感知功率。

后面就是加上11p的影响，但是我们这里不考虑，所以就不再赘述。

然后将那些当前功率小于噪声功率的抹零，因为需要避免分配过程中这种小干扰的影响。

sensedPowerCurrentSF\_MHz(sensedPowerCurrentSF\_MHz<phyParams.Pnoise\_MHz) = 0;

然后更新一次感知矩阵：

stationManagement.sensingMatrixCV2X(1,BRids\_currentSF,stationManagement.activeIDsCV2X) = sensedPowerCurrentSF\_MHz;

选择第一行是因为之前已经把最后一行即最后测量的值挪上来了。

接下来就是更新已知的使用矩阵(**knownUsedMatrix**)，它的作用是从SCI消息中读取状态消息。首先需要知道它的每一维指的是什么，它的初始化是在mainInit.m中进行的:

stationManagement.knownUsedMatrixCV2X

= zeros(appParams.Nbeacons,simValues.maxID);

=zeros(500,200);

然后开始循环更新每辆车和每个BR的已知使用矩阵：

stationManagement.knownUsedMatrixCV2X(BRids\_currentSF,:) = 0;

但是因为此时没有正在使用V2X技术传输的车辆，那么这个函数有一部分会先跳过，这个等下再来讨论，接下来就会弹出到mainCV2XttiEnds.m。接下来就会运行下一个函数BRreassignment3GPPautomous.m函数。

进入这个函数，首先就是检查是否需要进行重选。

首先确定下一个资源所在的子帧，以及给下一次分配所使用的子帧。

subframeNextResource

= ceil(stationManagement.BRid/appParams.NbeaconsF);

subframesToNextAlloc

=zeros(length(stationManagement.BRid(:,1)),phyParams.cv2xNumberOfReplicasMax);

其中stationManagement.BRid的生成是采用了它的算法101（Random Allocation）这样就得到了一个随机的BRid序列，范围为[1,500]。就是这200辆车所随机选择的BRid。所以它的维度是200×1。

下一次分配的子帧的维度也是200×1，因为重传目前只支持一次，所以为1。

然后会进入一个条件判断函数：

allConditionsMet

= false(length(activeIDsCV2X),phyParams.cv2xNumberOfReplicasMax);

subframesToNextAlloc(:,j)

= (subframeNextResource(:,j)>currentT).\*(subframeNextResource(:,j)-currentT)

+(subframeNextResource(:,j)<=currentT).\*(subframeNextResource(:,j)

+appParams.NbeaconsT-currentT);

上面这个式子表示的是下一个分配的子帧是哪个，比如currenT为24，而下一个资源的子帧为12，那么它的对应值就是12。

需要判断A B C D四个条件，它们分别为：

A = stationManagement.cv2xNumberOfReplicas(activeIDsCV2X) >= j;

B = stationManagement.BRid(activeIDsCV2X,j)>0;

C =

(subframesToNextAlloc(activeIDsCV2X,j) < simParams.T1autonomousModeTTIs | subframesToNextAlloc(activeIDsCV2X,j)>simParams.T2autonomousModeTTIs);

D = false(当j=1时就为false，而目前只能是1)

A目前只能是True，B也都是True，C目前都是False。

那么

allConditionsMet(:,j) = A & B & (C | D) ;

= 1&1&0

=0

（所以上面这一部分是干啥的，我目前还不清楚）

接下来就是检验车辆的下一次分配是否会在此时的slot上进行资源分配：

hasFirstResourceThisTbeacon

= (subframeNextResource(activeIDsCV2X,1)==currentT);

然后更新重选计数器Reselection Counter（RC）：

stationManagement.resReselectionCounterCV2X(activeIDsCV2X) =

stationManagement.resReselectionCounterCV2X(activeIDsCV2X)-hasFirstResourceThisTbeacon;

如果hasFirstResourceThisTbeacon等于1即RC=RC-1。

然后当RC=1时需要去判断是否需要进行重选步骤，如果它不需要重新选择新的资源的话，那么RC将会在RC=0之前再更新一次RC。