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。

```
appParams:
  包含以下字段的 struct:

                    allocationPeriod: 0.1000
          variabilityGenerationInterval: 0
                   generationInterval: 0.1000
    generationIntervalAverageRandomPart: 0
                     beaconSizeBytes: 350
                       resourcesV2V: 100
                     beaconSizeBits: 2800
                          RBsBeacon: 10
                     RBsFrequencyV2V: 52
                          NbeaconsF: 5
                          NbeaconsT: 100
                           Nbeacons: 500
                          nPckTypes: 1
                              nRSUs: 0
```

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。

μ	$\Delta f = 2^{\mu} * 15$	Frequency Range	Cyclic Prefix	Slot duration (ms)	# Slots/Subframe	n. PRBs in 10 MHz
0	15	sub-6 GHz	Normal	1	1	52
1	30	sub-6 GHz	Normal	0.5	2	24
2	60	Any	Normal, Extended	0.25	4	11
3	120	mmWave	Normal	0.125	8	NA

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。