## 2022/11/24 测试日志

测试目的：由2022/11/23测试日志引申对**mainV2X.m**进行代码运行分析，主要是观测**idEvent**、**timeEvent**中的一些判断标志作用。

### 测试1

这里设置的断点在mainV2X.m的时间循环里，即：

while timeManagement.timeNow < simParams.simulationTime

…

end

先找到最小的时刻作为timeEvent：

[timeEvent, indexEvent]

= min(timeManagement.timeNextEvent(stationManagement.activeIDs));

timeManagement.timeNextEvent(stationManagement.activeIDs)是一个200\*1的矩阵，它存储着200个车辆下一个事件的发生时间，我们需要按照时间顺序选择最早发生的。

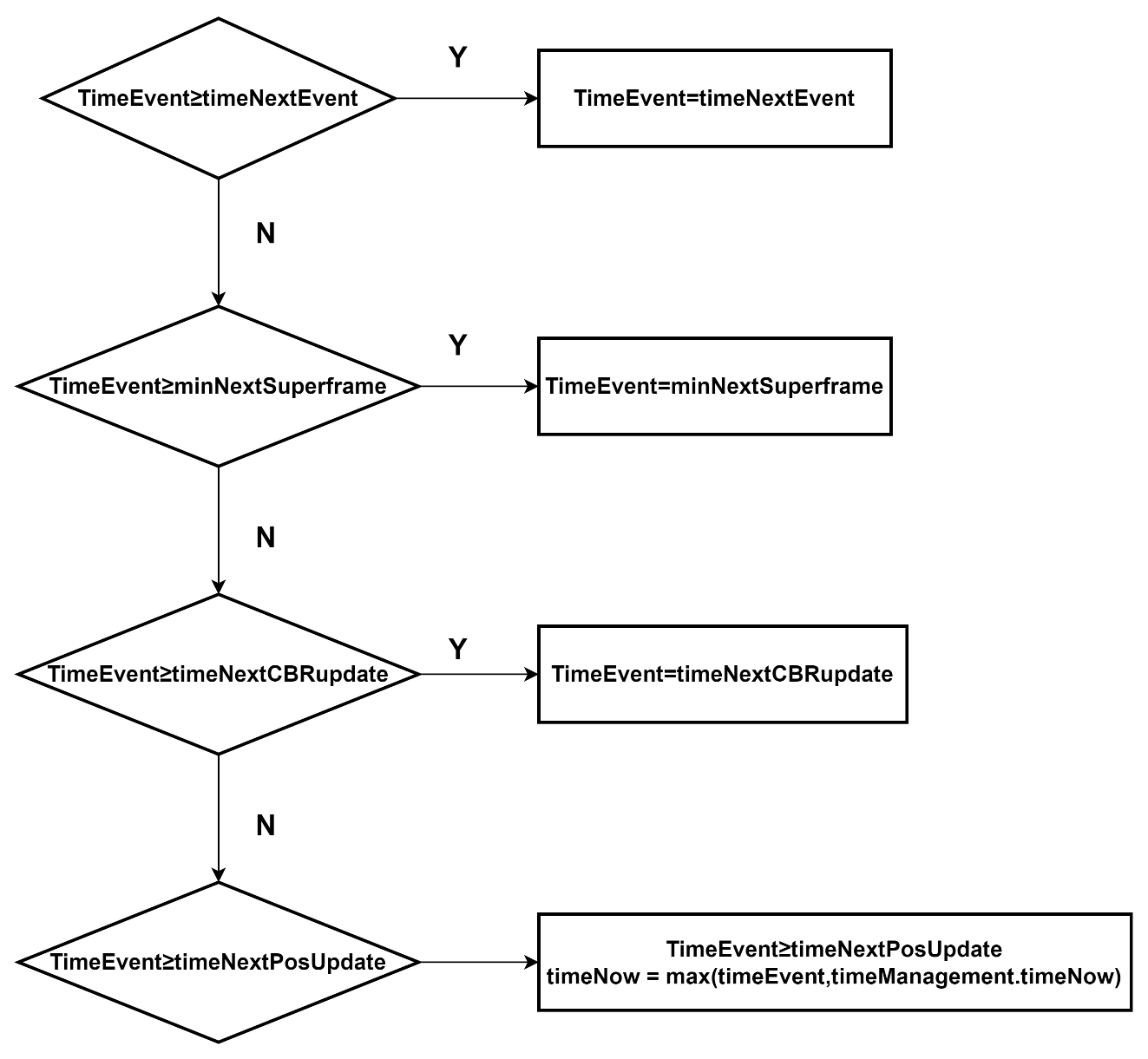
idEvent = stationManagement.activeIDs(indexEvent);

然后将发生事件的车的标号记录下来即idEvent。比如我这次的实验记录到了第159辆车是最早发生的，所以idEvent=159，此时timeEvent=0.029s。

而后需要用一下相关时间点的Event与timeEvent进行比较然后对timeEvent进行修正，分别是以下变量：

|  |  |
| --- | --- |
| 变量 | 属性 |
| timeManagement.timeNextCV2X | C-V2X Event发生事件时间 |
| minNextSuperframe | Superframe发生时间（即共存和方法A） |
| timeManagement.timeNextCBRupdate | CBR更新事件时间 |
| timeManagement.timeNextPosUpdate | 下个位置更新事件的时间 |

如果TimeEvent大于任何一个上述变量，则会将timeEvent修正为上述变量的值，流程图如下：



如果timeEvent小于timeNextPosUpdate,那么就会将：

timeManagement.timeNow = timeEvent;

后面还需要判断是否大于仿真时间，比如我们设置的是10s，等于10s后需要break跳出来。

而且每过0.1s就需要将时间信息打印在video即控制台界面。

接下来就是Action阶段，根据timeEvent的值进行对应的动作,就比如此时的timeEvent=0.029s,现在进行判断。

timeManagement.timeNextPosUpdate=0.100s, 说明此时它还不能执行位置更新函数mainPositionUpdate()。

timeManagement.timeNextCBRupdate=0.030s,说明此时它还不会执行

cbrUpdateCV2X()函数。

而minNextSuperframe=Inf，所以也不会执行superframeManagement()函数。

timeManagement.timeNextCV2X=0.029s,所以会执行这个部分的内容，通过判断标志位timeManagement.ttiCV2Xstarts观察执行mainCV2XttiStarts()函数还是mainCV2XttiEnds()函数。这两个函数是交替进行的，不会同时进行。这两个步骤一完成其实就是用了一个TTI的时间，从表达式中也可以看出，完成前一个函数后：

timeManagement.timeNextCV2X

= timeManagement.timeNextCV2X + (phyParams.TTI - phyParams.TsfGap);

完成后一个函数后：

timeManagement.timeNextCV2X

= timeManagement.timeNextCV2X + phyParams.TsfGap;

总体上的时间就是phyParams.TTI=0.001s。

此时timeEvent=0.029s,而此时timeManagement.timeNextPacket(idEvent)

=0.029s,所以也会执行这一部分。（这一部分指一个新的数据生成）。如果是LTE节点，则会执行bufferOverflowLTE()函数。

执行完之后，又回到循环开始，此时找到的最小timeEvent=0.054，idEvent=178。但是因为timeNextCV2X还是为0.029，所以，timeEvent被修正为0.029s。因为之前得timeNextCV2X=0.029s时完成了mainCV2XttiStarts()函数，这次0.029s是去完成mainCV2XttiEnds()函数。完成后，timeManagement.timeNextCV2X=0.03s,

这时候timeEvent == timeManagement.timeNextCBRupdate符合则执行这一部分即处理CBR的计算。

## 2022/11/25 测试日志

### 测试1

测试1的内容主要是对2022/11/24的测试日志中mainV2X.m中的timeEvent做一次相关性的总结，本次实验是对NR-V2X的中的一些发生时间做一个归纳性整理。首先下列所有的结果都是建立在使用了”**test.cfg**”的配置文件。

首先minNextSuperframe这个事件发生时间我们不予考虑，因为是用不到。

**C-V2X Event发生时间**

而对于timeManagement.timeNextCV2X的变化主要是根据mainCV2XttiStarts()和

mainCV2XttiEnds()函数来区分，如果是后者：

timeManagement.timeNextCV2X

= timeManagement.timeNextCV2X + phyParams.TsfGap;

= timeManagement.timeNextCV2X + 一个slot中一个OFDM符号的长度；

= timeManagement.timeNextCV2X + 1ms/14

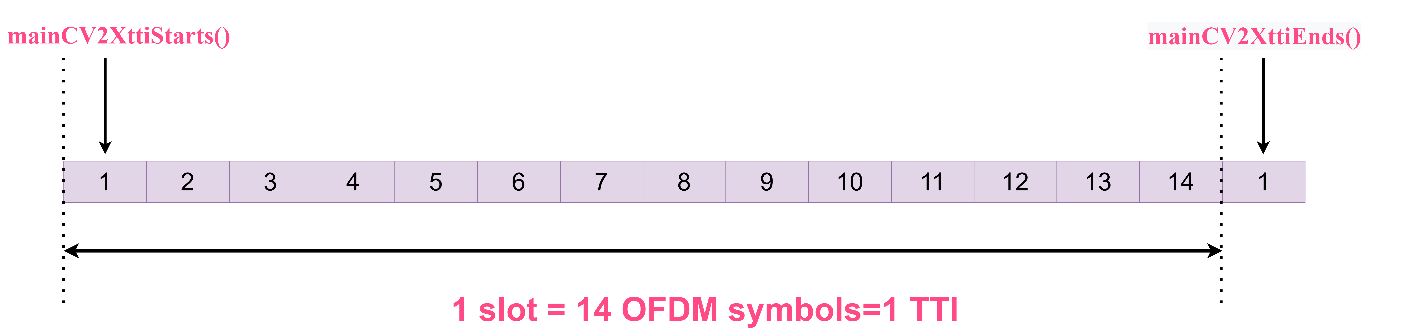
= timeManagement.timeNextCV2X+ 7.1429\*10e-5 s;

对于前者：

timeManagement.timeNextCV2X

= timeManagement.timeNextCV2X + (phyParams.TTI - phyParams.TsfGap);

= timeManagement.timeNextCV2X + (0.001+0.001/14);



然后我们还需要去关注一些事件的发生点，接下来会一一讲解，这样就知道什么节点发生什么事情：

**CBR更新时间：**

判断条件：

timeEvent == timeManagement.timeNextCBRupdate

timeManagement.timeNextCBRupdate的时间间隔为：

timeManagement.timeNextCBRupdate

= timeManagement.timeNextCBRupdate

+ (simParams.cbrSensingInterval/simParams.cbrSensingIntervalDesynchN);

它的变化时间为[0,1,2,3,4…]ms。

位置更新事件事件：

0.1s 0.2s 0.3s…

新数据产生事件事件：

这个是**2022/11/25 测试2**进行的。在那里具体讲解。

### 测试2

测试的是关于mainV2X.m中出发产生新数据部分的函数测试，判断语句为：

elseif timeEvent == timeManagement.timeNextPacket(idEvent)

此时timeEvent = 0.082s, idEvent = 43, 而等式右边的值恰好等于0.082s即：

timeManagement.timeNextPacket(idEvent)=0.082s。

然后我们只考虑LTE节点，首先会执行一个：

stationManagement.pckBuffer(idEvent) = stationManagement.pckBuffer(idEvent)+1;

关于stationManagement.pckBuffer的初始化在mainInit.m中这样初始化：

stationManagement.pckBuffer = zeros(simValues.maxID,1);

表示的是每个节点队列中的数据包数。这个很好理解，就是当在这个idEvent产生一个数据包，它就在这个节点的数据包数量加1。

如果此时这个节点的数据包数量大于1，则执行bufferOverflowLTE()函数，该函数的作用是（不是太懂），然后再令：

stationManagement.pckNextAttempt(idEvent) = 1;

因为NR增加了非周期性流量那么它的生成时间间隔变为了：

generationInterval =

timeManagement.generationIntervalDeterministicPart(idEvent)+exprnd(appParams.generationIntervalAverageRandomPart);

即常数部分加一个指数分布的随机数，这里我们暂时不考虑非周期，按照周期考虑。

然后将generationInterval与dcc\_minInterval进行比较，将较大的作为时间间隔，然后与当前时间相加作为下一次的生成数据时间。

此外，再更新一下：

timeManagement.timeLastPacket(idEvent)

= timeManagement.timeNow-timeManagement.addedToGenerationTime(idEvent);

### 测试3

通过2022/11/25 测试2可以得知，经过这个新数据产生的步骤将指定的最早发生的idEvent中的timeManagement.timeNextEvent进行更新，加上了一个生成间隔，接下来下一次循环的时候，就会重新选择新的idEvent所对应的timeEvent。此时最小的timeEvent就更新了，然后与上面所提到的Event时间进行比较。

## 2022/11/27 测试日志

### 测试1

本次测试主要是针对mainV2X.m中的位置更新进行分析，执行的函数是

mainPositionUpdate()

触发的条件是：

if timeEvent >= (timeManagement.timeNextPosUpdate-1e-9) && ...

(isempty(stationManagement.activeIDsCV2X) || timeEvent > timeManagement.timeNextCV2X || timeManagement.ttiCV2Xstarts==true)

timeManagement.timeNextPosUpdate的变化范围是[0, 0.1, 0.2, 0.3…]。

## 总结

经过这几天的测试完成了对WiLabV2Xsim的进一步测试，了解了

mainV2X.m的具体工作流程，这次对该函数的了解比之前更加清楚一些其中的细节和工作机制，接下来的工作就是