**CSMA/CD协议模拟仿真**

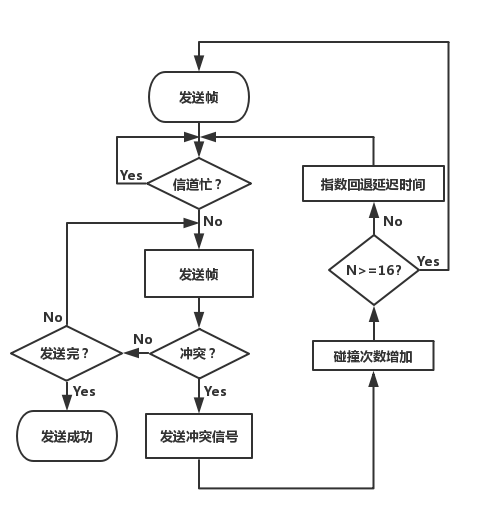
## 主要工作原理

1. 载波监听：任一用户节点(或站点)需要发送数据帧时，首先监测信道，用以确定传输介质上是否有其他用户节点正在发送信号。如果信道忙，则一直监测信道，直至空闲。
2. 冲突检测：由于数据帧在信道中传输时存在时延，也可能存在两个及以上用户同时发送数据，从而造成信道冲突，因此，每个站点在发送帧期间，需同时监测是否发 生冲突。一旦检测到冲突出现，就立即停止发送，并在信道上发送一串干信号 ，通 知发送方信道出现冲突。
3. 随机回退：用户节点检测到冲突后，为了降低再次发生冲突的概率，必需等待一个 随机时间，然后进行数据帧重传。这个随机时间采用用截断二进制指数回退算法。

## 建模

（1）语言选择：python（建议使用c或c++，struct或class比较适合存放一个站点的多项信息，并且能够在做一些比较或运算时整体移动，我发现这点时已经晚了。）

（2）仿真流程：



（3）程序：

首先有N个节点，时间为t=0，反复t+=1，争用期T1，帧长T2，设置7个list来储存N个节点的各项数据,。【0index，1状态（0空闲，1发送，2处于回退），2发送时序（如果状态为发送，每个时间t增加一），3监听位，4冲突位，5时序,6上次碰撞时间,7重传次数】（这7个list简称n1-n7）.

3.1规则

1. 如果某两个节点的n2之和大于T1/2，说明碰撞。
2. 如果n1==2，说明处于回退状态，要等时间到了才能够发送。
3. 只有（n1==0 and n3==0 and n4==0），节点才可以按照概率决定是否发送帧
4. 当存在n2==T2时，说明发送成功，输出循环

3.2函数

由于使用多个list，不方便把多个list当做哈数参数传输（应该使用c或c++里的struct或class），所以主要过程并没有使用函数，只是在增加时间时使用了函数def addtime(num)、去是否发送的概率时使用了函数def random\_pick(some\_list,probabilities)

1. addtime(num)：num里储存了此时n2==1的节点，也就是哪些节点的发送时序需要加。
2. random\_pick(some\_list,probabilities)：some\_list储存多个数，probabilities储存对应概率

比如some\_list=[0,1], probabilities=[0.2,0.8]，则这个函数有20%的概率返回0

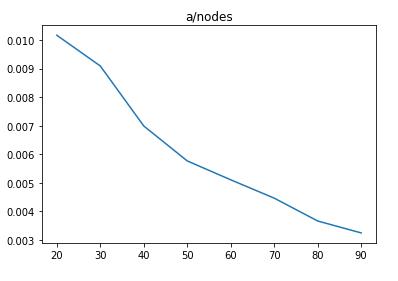
1. 二进制指数回退算法：int(random.randint(1, 2\*\*(min(n7[a],10))))

n7为重传次数，min()函数取两者最小值，random.randint(a,b)函数随机产生a-b之间的一个数，int()函数把float型转成int型

## 结论

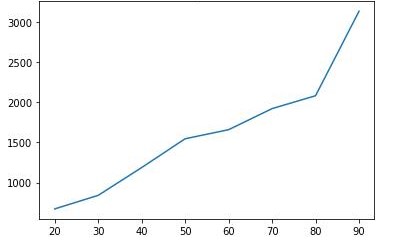
（下图的每个点均是多次计算取平均值）

1. 参数a(y轴)与节点数量nodes(x轴)(T1=20，T2=30）



可以发现，随着节点数量的增加，系统发送失败的概率越大，导致帧的期望发送时间越大，在争用期不变的情况下，参数a逐渐下降。

1. 成功发送一帧所需要的时间(y轴)与争用期T1(x轴)



可以发现，争用期越大，系统发送失败的概率越大，导致成功发送一帧所需要的时间也逐渐增大。