调研情况

TDMA-VANET 要解决的问题是两个：join collision 和 merge collision。在 VANET 中，由于车辆运动快，merge collision 的影响会比较大。根据最新的文献，主流的做法是考虑车辆的运行方向，将一帧分为左右两部分；不同方向的车辆会到不同的部分去申请时隙以避免 merge collision。但这个做法一个不好的地方就是不能很好的适应车流不均衡的情况（不同的方向车辆密度差距大），为了解决这个问题，一些工作试图通过引入更多的判断信息来动态规划不同方向车辆可以申请的时隙数量。在这些工作中，时隙分配的基础依然是 RR-ALOHA。在其中，每个节点在一帧开头时广播当前帧的时隙占用情况（FI 报文）；每个节点收到 FI 后就能够直接知晓两跳范围内所有节点的时隙占用情况。每个节点需要保证的是自己所选择的时隙在两跳范围内的节点中是独占的。

这些工作的一个共同点是使用 2 跳邻居信息判断时隙占用情况，这种设计无法对已经占用的 BCH 进行调整，在高动态的车载网环境下是不合理的。

本方案简述

为了实现 BCH 可调这一目的，本算法在每个节点广播 FI 的基础上广播在每个时隙上，两跳邻居的占用的数量（这个信息是由原 FI 报文直接累加出来的）。根据增强版的 FI 报文，节点可以将收到的“两跳节点占用情况 $Count_{2hop}$”（每个时隙占用 8 位）进行累加，计算出每个时隙的三跳邻居占用数量 $Count_{3hop}$。算法根据 $Count_{3hop}$ 的值来对每一个时隙进行评价，$Count_{3hop}$ 的值越低代表该时隙被隐藏节点所占用的可能性更小。由于存在重复计算，$Count_{3hop}$ 的值会比实际情况要偏大，但由于算法只进行相对值的对比，所以偏大的值实际上不会对时隙评价产生影响。有了对每个时隙的评价后，节点会选择评价最优（根据 $Count_{3hop}$）的两跳邻居未占用（根据原始 FI）时隙作为自己的基础时隙（BCH）。在选择好基础时隙后，节点可以根据时隙评价的变化情况实时对自己的 BCH 做出调整，以此来最大程度避免时隙冲突。以下是时隙分配的具体设计。

控制信道的时隙分配

控制帧种类与格式：REQ、FI、ADJ、BAN

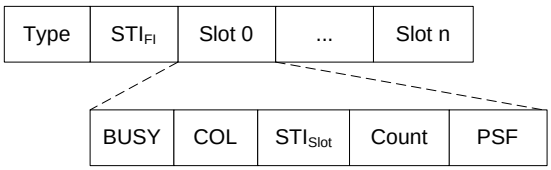
帧格式

REQ 帧格式：

Type	STI_{REQ}	PSF_{REQ}	$Slot_{REQ}$

- STI_{REQ} : Short ID of the request sender (8 bits).
- BUSY n : If Slot n is busy or free (1 bit).

FI 帧格式：



- STI_{FI} : Short ID of the FI sender (8 bits).
- BUSY and COL: Shows status of slot n (1 bit, respectively).
- STI_{Slot} : Short ID of the user of slot n (8 bits).
- Count : The number of two-hop neighbors Of the slot n user (8 bits).
- PSF : Priority (2 bits).

ADJ 帧格式：

Type	STI_{ADJ}	PSF_{ADJ}	$Slot_{ADJ}$

- STI_{ADJ} : Short ID of the request sender (8 bits).
- **ET : Effective time of this request (8bits).
- PSF_{ADJ} : Priority of this request (8 bits).
- $Slot_{ADJ}$: Target slot number (8 bits).

BAN 帧：

Type	STI_{BAN}	PSF_{BAN}	$Slot_{BAN}$

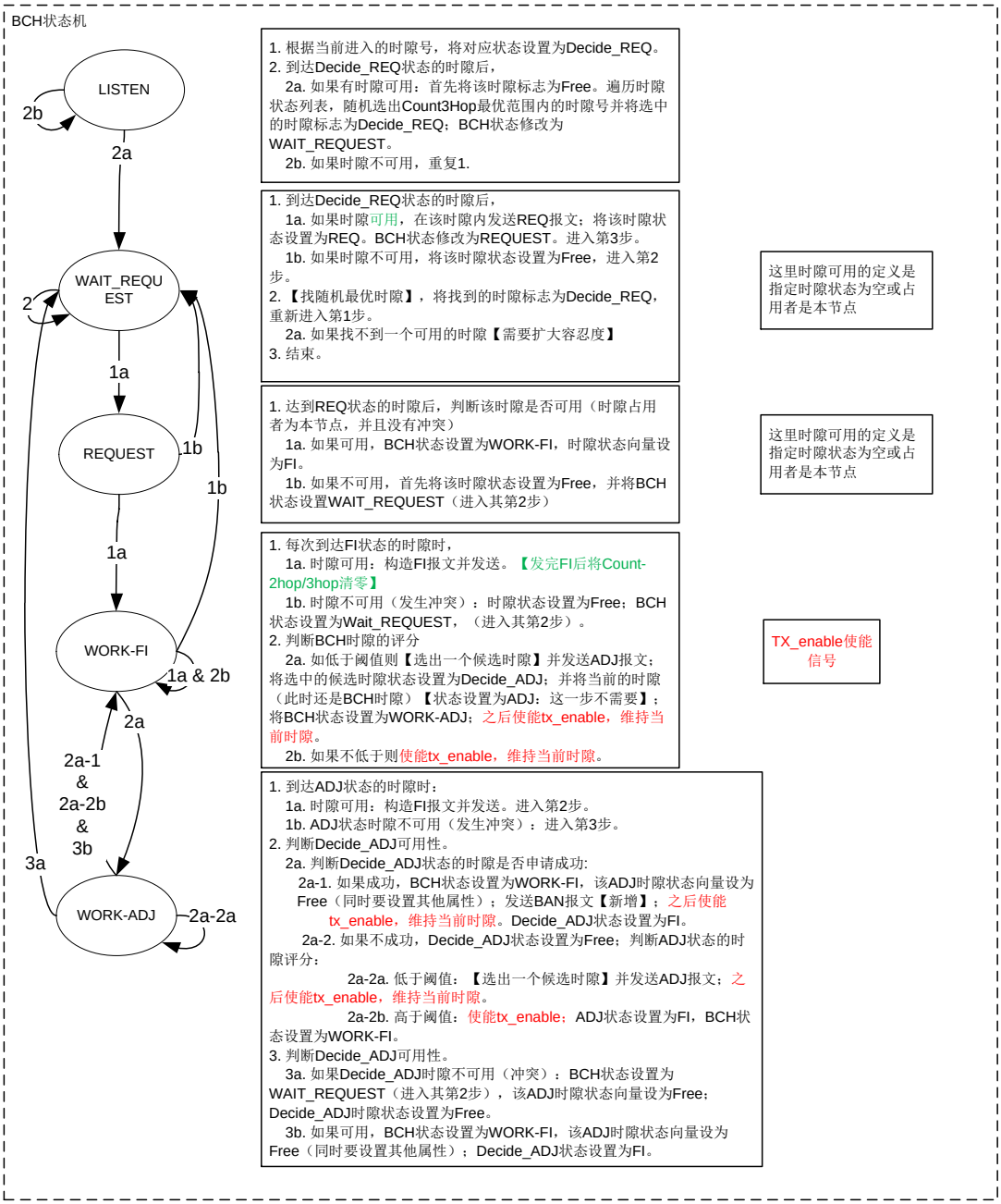
- STI_{BAN} : Short ID of the request sender (8 bits).
- PSF_{BAN} : Priority of this request (8 bits).
- $Slot_{BAN}$: Target slot number (8 bits).

状态维护

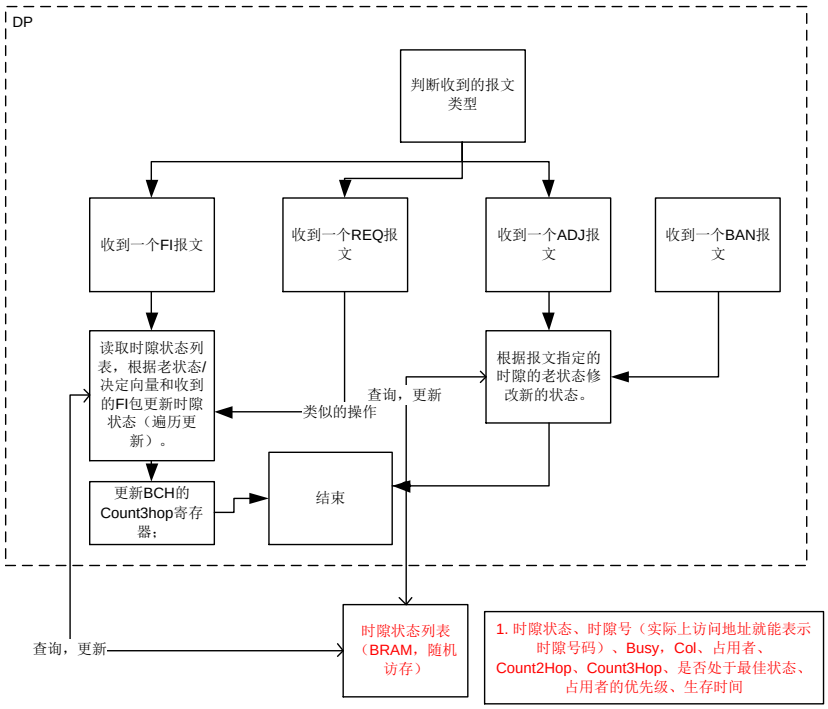
需要维护两种状态：（1）节点的 BCH 状态，以及（2）每个时隙的状态。
针对每个时隙状态，对收到的控制报文进行处理，并更新时隙情况列表。

BCH 时隙分配状态机

下面的状态机描述的 BCH 的申请与维护状态机。


BCH状态机
LISTEN
2b
2a
WAIT_REQUEST
2
1a
REQUEST
1b
1b
1a
WORK-FI
1a & 2b
2a
2a-1
&
2a-2b
&
3b
3a
WORK-ADJ
2a-2a
1. 根据当前进入的时隙号，将对应状态设置为Decide_REQ。
2. 到达Decide_REQ状态的时隙后，
2a. 如果有时隙可用：首先将该时隙标志为Free。遍历时隙状态列表，随机选出Count3Hop最优范围内的时隙号并将选中的时隙标志为Decide_REQ；BCH状态修改为WAIT_REQUEST。
2b. 如果时隙不可用，重复1.
1. 到达Decide_REQ状态的时隙后，
1a. 如果时隙可用，在该时隙内发送REQ报文；将该时隙状态设置为REQ。BCH状态修改为REQUEST。进入第3步。
1b. 如果时隙不可用，将该时隙状态设置为Free，进入第2步。
2.【找随机最优时隙】，将找到的时隙标志为Decide_REQ，重新进入第1步。
2a. 如果找不到一个可用的时隙【需要扩大容忍度】
3. 结束。
这里时隙可用的定义是指定时隙状态为空或占用者是本节点
1. 达到REQ状态的时隙后，判断该时隙是否可用（时隙占用者为本节点，并且没有冲突）
1a. 如果可用，BCH状态设置为WORK-FI，时隙状态向量设为FI。
1b. 如果不可用，首先将该时隙状态设置为Free，并将BCH状态设置WAIT_REQUEST（进入其第2步）
这里时隙可用的定义是指定时隙状态为空或占用者是本节点
1. 每次到达FI状态的时隙时，
1a. 时隙可用：构造FI报文并发送。【发完FI后将Count-2hop/3hop清零】
1b. 时隙不可用（发生冲突）：时隙状态设置为Free；BCH状态设置为Wait_REQUEST，（进入其第2步）。
2. 判断BCH时隙的评分
2a. 如低于阈值则【选出一个候选时隙】并发送ADJ报文；将选中的候选时隙状态设置为Decide_ADJ；并将当前的时隙（此时还是BCH时隙）【状态设置为ADJ：这一步不需要】；将BCH状态设置为WORK-ADJ；之后使能tx_enable，维持当前时隙。
2b. 如果不低于则使能tx_enable，维持当前时隙。
TX_enable使能信号
1. 到达ADJ状态的时隙时：
1a. 时隙可用：构造FI报文并发送。进入第2步。
1b. ADJ状态时隙不可用（发生冲突）：进入第3步。
2. 判断Decide_ADJ可用性。
2a. 判断Decide_ADJ状态的时隙是否申请成功:
2a-1. 如果成功，BCH状态设置为WORK-FI，该ADJ时隙状态向量设为Free（同时要设置其他属性）；发送BAN报文【新增】；之后使能tx_enable，维持当前时隙。Decide_ADJ状态设置为FI。
2a-2. 如果不成功，Decide_ADJ状态设置为Free；判断ADJ状态的时隙评分：
2a-2a. 低于阈值：【选出一个候选时隙】并发送ADJ报文；之后使能tx_enable，维持当前时隙。
2a-2b. 高于阈值：使能tx_enable；ADJ状态设置为FI，BCH状态设置为WORK-FI。
3. 判断Decide_ADJ可用性。
3a. 如果Decide_ADJ时隙不可用（冲突）：BCH状态设置为WAIT_REQUEST（进入其第2步），该ADJ时隙状态向量设为Free；Decide_ADJ时隙状态设置为Free。
3b. 如果可用，BCH状态设置为WORK-FI，该ADJ时隙状态向量设为Free（同时要设置其他属性）；Decide_ADJ状态设置为FI。

收到控制报文的处理：



FI 报文的处理以及时隙状态的维护

表． MARR-ALOHA 时隙描述

信道状态	(Busy1, Busy2)	STI_{slot}的取值
FI 发送者占用	(1,0)	FI 发送者编号
直接邻居占用	(1,0)	时隙占用者编号
2 跳邻居占用	(0,1)	-
2 跳内无占用	(0,0)	-

收到 FI 报文的处理

收到一个 FI 报文后，需要根据报文中描述的每个时隙信息更新本节点维护的时隙信息。对于一个时隙的占用情况有四种：自己占用；直接邻居占用；两跳邻居占用；空闲。以下根据四种情况分别讨论时隙状态的迭代。

【自己占用】：时隙状态向量为 REQ、FI、ADJ、Decide_ADJ 【这里也包含了申请后判断是否可用】

1. 别人发的 FI 认为这个时隙不是我的（STI-slot-local ！= STI-slot）（他自己占的；他邻居占的）【异常情况】
    a) FI 是（1,0），不管是他占用的还是他邻居占用的，表明两跳内肯定出现了冲突。此时要判断优先级，他的优先级比我的优先级低或等于，则我保持时隙信息不变；若高于，则将 STI-slot-local 重新赋值为 STI-slot；时隙状态向量改为 Free。若是他占用的，状态维持（1,0）不变；如果是他邻居占用的，状态改为（0,1）。
    b) FI 是（0,1），不一定产生了冲突，维持原有状态。
2. 别人发的 FI 认为这个时隙是我的。（可以是（1,0）或（0,1））。
3. （0,0）：这个 FI 发送节点必须是新邻居，否则出现协议错误。

==========下面三种情况都处于向量 Free、Decide_REQ ============

【直接邻居占用】：

1. 别人发的 FI 和我掌握的情况一致（STI-slot-local == STI-slot）（可以是（1,0）或（0,1））。状态不变。
    a) 若 STI-slot == STI-FI （说明这个时隙是 FI 发送者占用的）则刷新该时隙的【生存时间】。生存时间可以设置为几个帧长的时间，用于节点主动退网。
2. 别人发的 FI 和我掌握的情况不一致（STI-slot-local ！= STI-slot 或 （0,0））
    a) （1,0）：不管是他占用的还是他邻居占用的，表明两跳内肯定出现了冲突。此时要判断优先级，他的优先级比我掌握的该时隙的占用着优先级低或等于，则我保持时隙信息不变；若高于，则将 STI-slot-local 重新赋值为 STI-slot。若是他占用的，状态维持（1,0）不变；如果是他邻居占用的，状态改为（0,1）。
    b) （0,1）：此时不一定产生了冲突（考虑直线排列），维持原有信息不变。
    c) （0,0）：这个 FI 发送节点必须是新邻居，否则出现协议错误。

【两跳邻居占用】：

1. 直接收到的 FI 说这是他的时隙且（STI-slot-local == STI-slot），则该时隙变为【直接邻居占用】
2. 收到的 FI 和我掌握的情况一致且不是他的时隙。
3. 收到的 FI 和我掌握的情况不一致（STI-slot-local ！= STI-slot 或 （0,0））【这里也会重复计算 count】
    a) 如果该 FI 是（0,1）则说明没有冲突：我维护的状态保持不变；
    b) 如果是（1,0），不一定冲突（考虑直线排列）：我维护的状态保持不变；
    c) 如果是（0,0），状态保持不变。

【空闲】

1. 收到的 FI 说这是他的时隙，则该时隙变为【直接邻居占用】；维护的状态从收到的 FI 处拷贝。
2. 收到的 FI 是（1,0）且不是他的时隙，则该时隙变为【两跳邻居占用】；维护的状态从收到的 FI 处拷贝。

3. 收到的 FI 是（0,1），则状态维持【空闲】。（因为最可能是两跳以外的节点了）。
4. （0,0）

时隙的释放

时隙的释放：存在三种情况：1. 节点退网；2. 发生冲突；3. 节点主动调整 BCH 到更优的时隙。

【节点退网】每个节点维护的“直接邻居占用时隙”的生存时间过期，则将该时隙的状态置 0.

【发生冲突】这种情况在 FI 的处理算法中进行了讨论，主要思路是优先级高的节点保持占用；优先级低的节点放弃占用。

【节点主动调整】节点主动调整时，会发送 BAN 报文。

生存时间的维护

直接邻居才需要记录其时隙生存时间；每次收到直接邻居的 FI 且 FI 发送者是该时隙占用者，则在其占用的时隙中更新其生存时间。生存时间以帧为单位，每次到达 BCH 时隙后需要对每一个维护了生存时间的时隙进行生存时间减一操作。

Count-2hop 和 Count-3hop 的维护

这两个计数器的维护是本算法的一个关键点，最基本的原则是 Count-2hop 不应该重复计算。但如果没有任何判断就对每个收到的 FI 信息进行累加的话会产生重复（考虑菱形的拓扑结构）。所以在这里需要定义几个规则避免 Count-2hop 重复累加。

1. 每次到自己的时隙发完 FI 后就清空所有非 BCH 时隙的 Count-2hop/3hop，每一帧重新维护。
2. 对于自己的直接邻居占用的时隙，只在时隙占有者发送的 FI 上累计 Count-2hop/3hop。
3. 对于自己的两跳邻居占用的时隙，需要进行记录，防止重复累计。
4. 对于自己占用的时隙，需要记录 Count-3hop

行驶方向的引入【目前还没有在 FPGA 上实现】

以上方案中没有使用到车辆的方向信息，我认为这是不合理的。存在如下的情况：假设有时隙 A 和时隙 B，他们的 $Count_{3hop}$ 值一致；有两个方向的车辆。假设实际占用时隙 A 的节点刚好都是由东向西，而占用时隙 B 的节点都是由西向东。则对于一辆由东向西的车辆来说，申请时隙 A 明显要比申请时隙 B 要好（因为方向一致，所以时隙的占用情况更加稳定，时隙碰撞的可能性也会降低）。这个问题是刘瑞霖方案中没有考虑的。

我的基本思路是将车辆方向信息加入两跳节点占用情况 $Count_{2hop}$ 中，节点可以分别计算一个时隙在两个方向的车辆的 $Count_{3hop}$；并给一个时隙两个评价值分别针对两个方

向。具体的做法是，将原 $Count_{2hop}$ 的 8 位拆成 4+4 位分别代表两个方向。要注意的是 4 位最大表示的数是 15，由于这个值是每个时隙都有的，我认为 15 应该是足够的。

【以下是服务时隙的设计，暂时未完善】加入的一些新设计

算法开销的降低

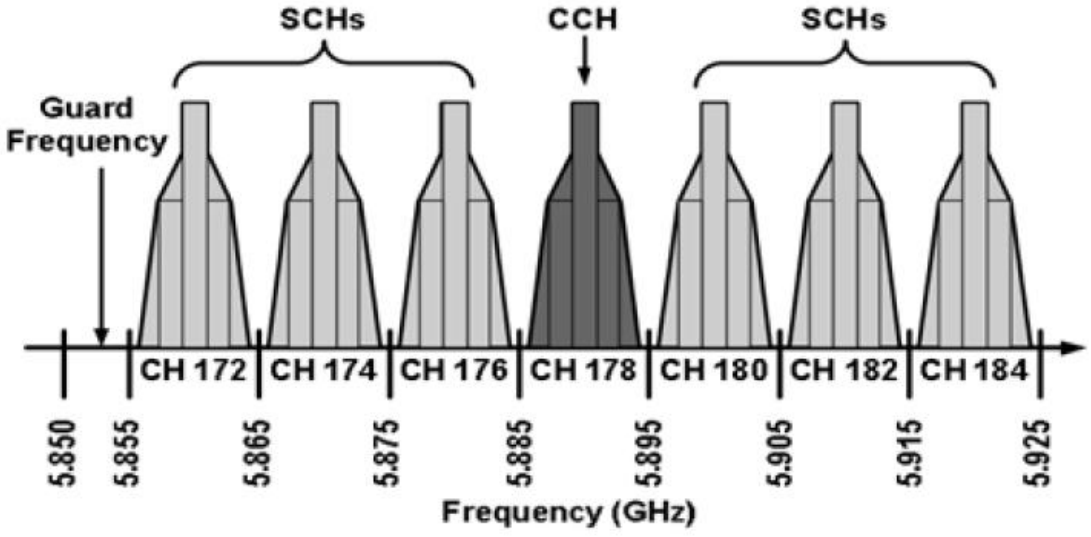
由于在 FI 的基础上额外引入了 8 位的 $Count_{2hop}$，这意味着算法开销的进一步扩大。当一帧为 100ms，包含 100 个时隙时，每个 FI 包的开销在 RR-ALOHA 的基础上增加 8*100/8 = 100 字节。则对于 1ms 一个的时隙来说，总开销约为 23 * 100 / 8 = 287.5 字节，开销额外增加 53 %。在之前的测试中，以 12M 的速度发 287 字节时间需要约 300us，则总开销占带宽的 30%以上。这是一个很可观的数字。

我的改进思路是将一个帧进一步分为两个小帧，每个节点只能选择其中一个小帧内的时隙进行申请；并且只需要广播当前小帧内所有时隙的占用情况。这样的设计会使算法开销减半。目前还没有想到会不会产生其他的负面效果。

多信道以及服务信道的接入

基本设计

整个车载网信道分为 7 个频点，包括 1 个 CCH（控制信道）和 6 个 SCH（服务信道）。



CCH 中时隙与帧长的设计：1 秒分为 1024 个时隙；每个时隙时长为 976.5625 us。1 秒分为 16 个帧，每个帧分为 64 个时隙（62.5 ms）。每个节点必须且只能在 CCH 上申

请一个时隙。CCH 负责发送 BCH 控制所需的 FI 包、SCH 接入的通知及确认报文、高优先级的安全报文。

SCH：采用帧长可变的设计，专门发送应用数据。应用数据分为“大量数据”和“实时数据（双工）”：大量数据的时隙申请策略为长帧、多时隙（连续），特点是延迟稍高、带宽大；实时数据的申请策略为短帧、单时隙，特点是低延迟，带宽小。

采用双接口设计：一个接口专门调至 CCH 频段；另一个接口专门用于 SCH 的收发（可以在多个 SCHs 上进行切换）。初期的实验可以用 433M+5.9G 组成双接口，用于演示少量节点时的情况；后期可以重新设计板子，做成两个 5.9G 模块的接口（原来的单 MiniPCIe 板子去掉几乎没用的 RJ45 接口和 USB 接口后应该可以很方便的放下另一个 MiniPCIE 插槽）。

CCH 的时隙接入由上面 BCH 分配策略进行控制，下面专门讨论 SCH 的接入。

SCH 的接入基本策略如下：

1. 由软件给出一个报文的属性（应用 ID，实时还是大量）；
2. 由硬件判断当前服务信道的占用情况，分析出最优的时隙和帧的分配情况；
3. FPGA 在 CCH 上进行通知（时隙和帧分配情况、服务信道号、下一跳节点编号），并直接占用时隙（无需再走申请-通过的流程）。
4. 每次的分配通知有效期为 CCH 的一帧长度。即每个节点在 CCH 帧中的时隙上通知接下来一个 CCH 帧时长中，其服务信道时隙占用情况和下一跳节点。

SCH 中根据占用情况和需求分配时隙

CCH 有 64 个时隙一个帧，最多可以容纳 64 个节点；每个节点最多只能在同一时间段申请一个 SCH 上的时隙（只有一个 SCH 接口）。即每个 SCH 信道最多会容纳 64 / 6 约 10 个点。

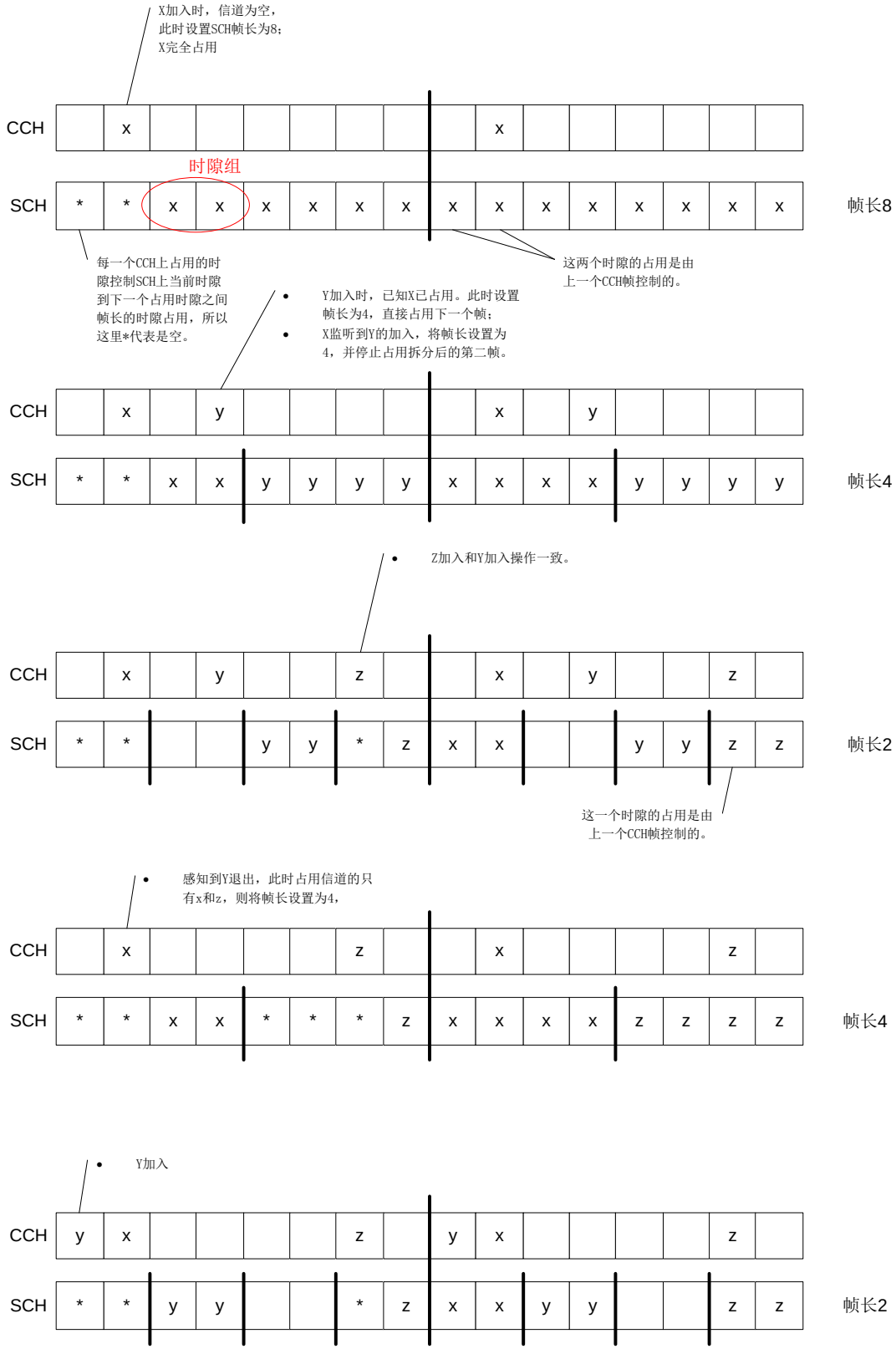
首先，服务信道的任务是提供点到点的帧传递（有或没有 Ack）。可以区别应对高数据量、延迟高和低数据量、延迟低的应用需求。

基本的 SCH 接入方式是直接占用、贪婪方式、根据新接入的节点实时调整时隙的占用。每个节点每次都直接占用“一个 SCH 帧”的长度，并且 SCH 帧长可以随着节点增加进行二分调整。一个 SCH 帧长最短为 2 个时隙，即每个节点分配到的时隙数量是 2 的倍数，我称成对的连续时隙为时隙组（由 A，B 两个时隙组成）；由此可以区分出两种数据方式的时隙占用：对于低延迟的应用，每个节点只占用时隙组的 A 时隙，留空 B 时隙给对向节点；对于大数据量的应用则每次都占满时隙组的 A，B 时隙，只在每一帧最后留一个时隙给对向节点用于 ACK 的传输（可选）。

以下是时隙分配示例（图片）：

其中存在的问题：

1. 怎样保证在有帧要发送时尽快发出去，并且对方还能够尽快收到（因为对方的服务接口不一定在这个频点上）。解决方法是可以从 CCH 中实时监听到节点 next-hop 目前所在的频点。有发送至 n 的帧倾向于直接使用 n 当前所在的频点；或是在 CCH 中通知 n 切换频点。这里又涉及到 n 节点是否在向其他节点传输数据的问题。
2. 潜在的碰撞问题：由于 CCH 的申请应该是避免了碰撞了，而 SCH 时隙的申请又是每一个 CCH 帧单独进行控制，所以 SCH 上应该不会出现碰撞的问题。


X加入时，信道为空，
此时设置SCH帧长为8；
X完全占用
CCH
SCH
时隙组
帧长8
每一个CCH上占用的时隙控制SCH上当前时隙到下一个占用时隙之间帧长的时隙占用，所以这里*代表是空。
这两个时隙的占用是由上一个CCH帧控制的。
Y加入时，已知X已占用。此时设置帧长为4，直接占用下一个帧；
X监听到Y的加入，将帧长设置为4，并停止占用拆分后的第二帧。
帧长4
Z加入和Y加入操作一致。
帧长2
这一个时隙的占用是由上一个CCH帧控制的。
感知到Y退出，此时占用信道的只有x和z，则将帧长设置为4，
帧长4
Y加入
帧长2

对实时数据和大量数据的时隙分开分配和维护。由于服务接口是在各个 SCH 之间切换，在同一个时间段内有多个不同需求的应用时，算法只能按优先级顺序满足。一共有 6 个 SCH，可以按顺序默认将 SCHs 的帧长分为设置为 2、4、8、16、32、64。【这里实际上可以不理解为帧长的调整。因为都是 2 的次幂，所以可以看做是统一的帧长 64，差别在于申请时隙的数量以及排列的方式。】

【这里主要有两个问题：一个是怎样保证在有帧要发送时尽快发出去，并且对方还能够尽快收到（因为对方的服务接口不一定在这个频点上）；第二个是怎样解决不均衡的负荷（全是实时数据或全是大量数据）】

1. 第一个问题：可以从 CCH 中实时监听到节点 next-hop 目前所在的频点。有发送至 n 的帧倾向于直接使用 n 当前所在的频点。

【数据类型的区分的结果是，一次申请多个时隙还是间隔申请时隙。很多数据混在一起时，需要保证高优先级也就是实时数据，即有实时数据时都采用间隔的时隙申请。（这里应该有一个切换的过程，即当有实时数据时，时隙采用间隔申请；没有实时数据只有大量数据时）好像也不需要切换的，因为可以采用每个控制时隙直接占用下一个数据时间片】

【基本的策略是贪婪策略，邻居节点多也不代表占用服务数据的节点多（）】【每个控制时隙直接占用接下来一个帧长度的数据时隙（贪婪占用），有新的节点需要占用服务时隙时，此时信道一定是满的；该节点直接发送占用请求（怎样保证公平？直接对半分怎么样？最多有 64 个节点的话每个频点最多有 10 个节点同时占用），。

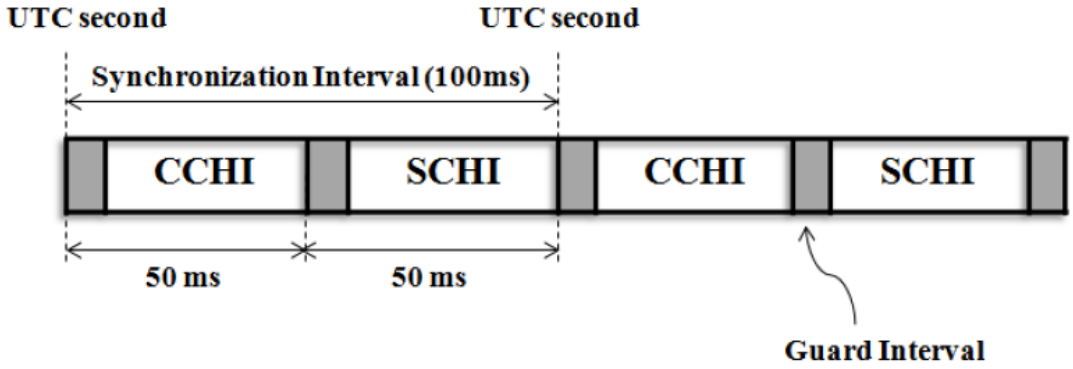
可以作如下假设：

~~对每个应用 ID 单独控制时隙的分配与维护~~。

1. 实时数据（双工）策略

1. 一个节点只申请一个时隙，尽可能保证帧长短。
2. 当一个节点没有
3. 在有信道完全空闲时，选取空闲信道，并按照单播还是多播设定帧长：单播的话帧长为 2 个时隙，当前节点 a 和下一跳节点 n 各占一个；当 n 在 CCH 上收到 a 的服务信道通知后
4. 根据新的申请调整已经分配的时隙

【也许软件可以做到很高精度，但实际上和负载程度密切相关：在高负载的情况下根本无法保证 TDMA 精度】



两种方案，一种是只用一个接口，每 50ms 切换控制信道和服务信道；***在车辆密度较大时以一定概率（密度越大概率越大）禁止车辆切换到服务信道，即将 SCHI 也拓展成为 CCHI

另一种方案是使用两个接口，一个始终对准 CCHI。

【从协议上来看这两种方式并没有太大的区别】

服务时隙的申请：每个节点在 CCHI 上报告其下一个 SCHI 时间片上占用的时隙数量（和所在的信道）；以及下一跳节点。【是否要考虑负载平衡，即根据当前信道的占用情况以及自己的负载大小来动态决定需要申请的时隙数量】

【分为两种情况进行讨论：单接口、双接口】：单接口的问题在于，每 50ms 需要进行一次切换。这个时候再讨论实时数据好像已经没有意义了。

我倾向于

实现 MARR-ALOHA 的步骤：

1. 在 dp 中收到 FI 包时对时隙列表进行更新【需要一套逻辑】，累加 count
2. Tc 中控制 BCH 状态机；时隙评分，并根据评分动态调整时隙。
    a) 依据刘的论文，有两种状态需要维护，分别是节点的 BCH 状态和每个时隙的操作状态（需要进行什么样的操作）