|  |  |
| --- | --- |
|  | **2016** |
|  | Tongji University |

|  |
| --- |
| **Unlicensed频谱中发射功率与信道空闲控制仿真报告** |
| Standalone unlicensed LTE (SAiL) 项目研究结果之三 |

目录

[第一章技术介绍 2](#_Toc446589561)

[调度和信道接入 2](#_Toc446589562)

[优先级别 5](#_Toc446589563)

[空闲信道评估 6](#_Toc446589564)

[许可辅助访问(LAA) 8](#_Toc446589565)

[静态空闲信道评估 9](#_Toc446589566)

[动态空闲信道评估 9](#_Toc446589567)

[发射功率控制 10](#_Toc446589568)

[评估方法 10](#_Toc446589569)

[第二章仿真结果与分析 13](#_Toc446589570)

[仿真程序源代码 14](#_Toc446589571)

[仿真结果（频率复用因子为1） 29](#_Toc446589572)

[仿真结果（频率复用因子为3） 34](#_Toc446589573)

[仿真结果对比（频率复用因子为1与频率复用因子为3的相同与不同） 39](#_Toc446589574)

[结论 39](#_Toc446589575)

[参考文献 40](#_Toc446589576)

[Figure 1 CSMA CA Mechanism Illustration. 3](#_Toc446586839)

# 第一章技术介绍

LTE系统：本文中，LTE的英文全称为Long Term Evolution，是3GPP（Third Generation Partnership Project，第三代合作伙伴计划）组织制定的UMTS（Universal Mobile Tele-communications System，通用移动通信系统）技术标准的长期演进，主要指3GPP LTE Release 8， Release 10, Release 12等第四代蜂窝移动通信系统中的通信技术。

WiFi系统：本文中，WiFi主要指以IEEE 802.11n, IEEE 802.11ac为代表的IEEE 802无线局域网通信技术。严格来说，WiFi是WiFi产业联盟的名称，具体技术名称应为WLAN (Wireless Local Area Network)。由于历史原因以及习惯原因，在本文中，WiFi与WLAN均指以IEEE 802.11n, IEEE 802.11ac为代表的IEEE 802无线局域网通信技术，包括物理层以及链路接入层（MAC）。

### 调度和信道接入

* LTE系统中，UE想要接收数据，需要先接收eNB发出来的downlink channel assignment。Downlink channel assignment一般通过Downlink control channel PDCCH发出来，告知UE在什么时刻、什么频段或资源块去接收下行数据，并告知UE数据发送出来的格式，比如码字个数、闭环closed loop MIMO、开环open loop MIMO、dual layer beamforming，多用户MIMO等等。根据PDCCH中的信息，UE会在指定时刻到指定资源块用指定格式去解调数据。在此过程中，UE会提供一定程度的反馈信息，如信道反馈，CQI反馈等汇报当前信道的基本情况。然而，最终的调度决定由eNB全权决定，即eNB可以完全根据UE的反馈来决定downlink channel assignment，也可以override UE的反馈自行决定downlink channel assignment。用户须遵从eNB的downlink channel assignment。

另一方面，LTE系统中，UE想要发送数据，需要先收到eNB发出来的uplink grants 。类似于下行情况，uplink grants一般通过downlink control channel PDCCH 发出来，告知UE在什么时刻、什么频段或资源块去发送上行数据。用户必须遵从 eNB的uplink grants，不可以自行选择什么时候、自行选择什么频段发送数据。

* 相对比，IEEE 802.11无线局域网采用的是一种分布式调度策略distributed coordination function DCF。IEEE 802.11系统中，每个发射机（上行链路中的STA或下行链路中的AP）均是自行调度来决定什么时候发数据。目前的IEEE 802.11系统（截止到IEEE 802.11 ac/ah）中，发送频段为整个系统频段。因此发射机不需要选择频段或资源块，直接在整个系统频段上发送信号。接收机可以提供一定程度的信道反馈，如channel state information feedback或者SNR feedback（类似于LTE系统中的CQI反馈）。

然而这些反馈仅用来给发射机做参考，发射机可以按照自己的算法来决定发射信号功率，调制解调方案等。由于每个发射机自行决定发送时刻，而各个发射机之间并不知晓别的发射机是否会同时进行发射操作，因此信道碰撞以及干扰必然发生。鉴于此，IEEE 802.11采用Carrier Sensing Multiple Access with Collision Avoidance基于碰撞避让的载波侦听技术来解决、缓解干扰问题。

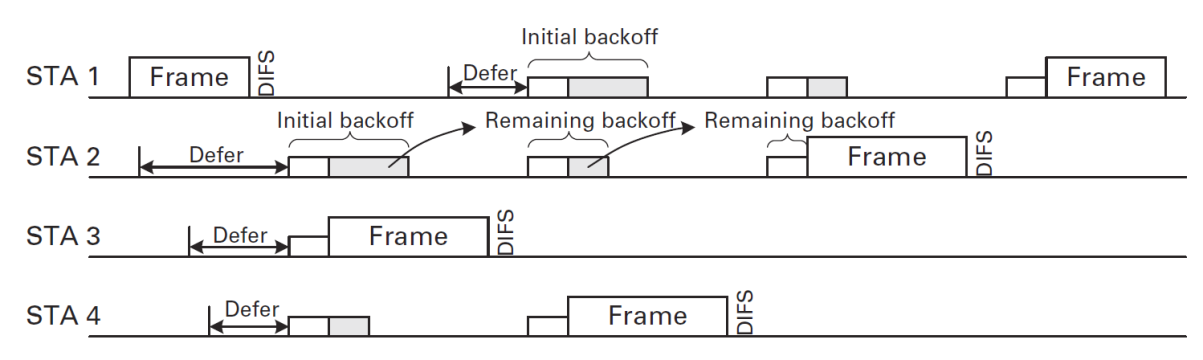


Figure 1 CSMA CA Mechanism Illustration.

如下图所示，STA 1 开始发送一个数据frame；同时STA 2，STA 3，STA 4进行载波侦听并都侦听到信道忙（STA 1 传输）；STA 2，STA 3，STA 4 分别选取不同的随机backoff 长度进行等待；STA 1 传输结束后，STA 3 选取的随机backoff等待最小并在三个STA中最先进行传输；此时STA 2， STA 4 冻结backoff计数器。直到STA 3传输结束DIFS以后，STA 2，STA 4继续原来的backoff；此后STA 4选取的随机backoff等待次小并在剩下的两个STA中先进行传输；此时STA 2仍然冻结backoff计数器直到STA 4传输结束DIFS后，获得信道接入并进行传输。无线局域网中的载波侦听起源于有限局域网中的载波侦听并很类似。不同于有线网碰撞检测，在无线局域网中只能采用碰撞避让的方式。

IEEE 802.11也定义了一种集中式调度策略即point coordination function PCF。然而实际情况下PCF的应用并不广。绝大部分系统采用的是DCF function。很多IEEE 802.11产品并没有实现PCF功能。因此我们在本文中不考虑PCF。

* SAiL：可预见的SAiL系统中因为需要和其他的无线局域网共存，因此需要掌握无线局域网中的CSMA/CA 规则，在一定程度下加以利用。另一方面，SAiL系统在一定程度下（如果eNB可以reserve信道一段足够长的时间）也可以保持原有的调度策略，即通过downlink channel assignment and uplink channel grants来确定用户的下行上行资源分配。另外，可以进一步研究LTE框架下新的离散式调度算法或协议。离散式调度算法的好处是可以给UE/STA更大的自由度，并且有利于降低端到端时延。

一般来说，可以认为SAiL系统绝大部分时间下工作和传统的LTE非常类似、即集中式调度，小部分时间（周期性或非周期性）用于允许离散式调度（即CSMA/CA）。如果SAiL系统大部分时间都用来和传统的IEEE 802.11系统一样来进行离散资源调度（CSMA/CA），则SAiL的效率必然大降低。

### 优先级别

* 由于LTE系统是一个以集中式调度为主的系统，因此用户的优先级别主要是由基站eNB来决定的。如果eNB采用proportional fairness为准则的调度算法时，则proportional fairness metric大的用户有更高优先级，也会更早获得资源调度，而proportional fairness metric小的用户有较低优先级，也因此会较晚获得资源调度。然而proportional fairness metric主要在eNB端计算，而各个用户UE自身可能并不知道或不关心。

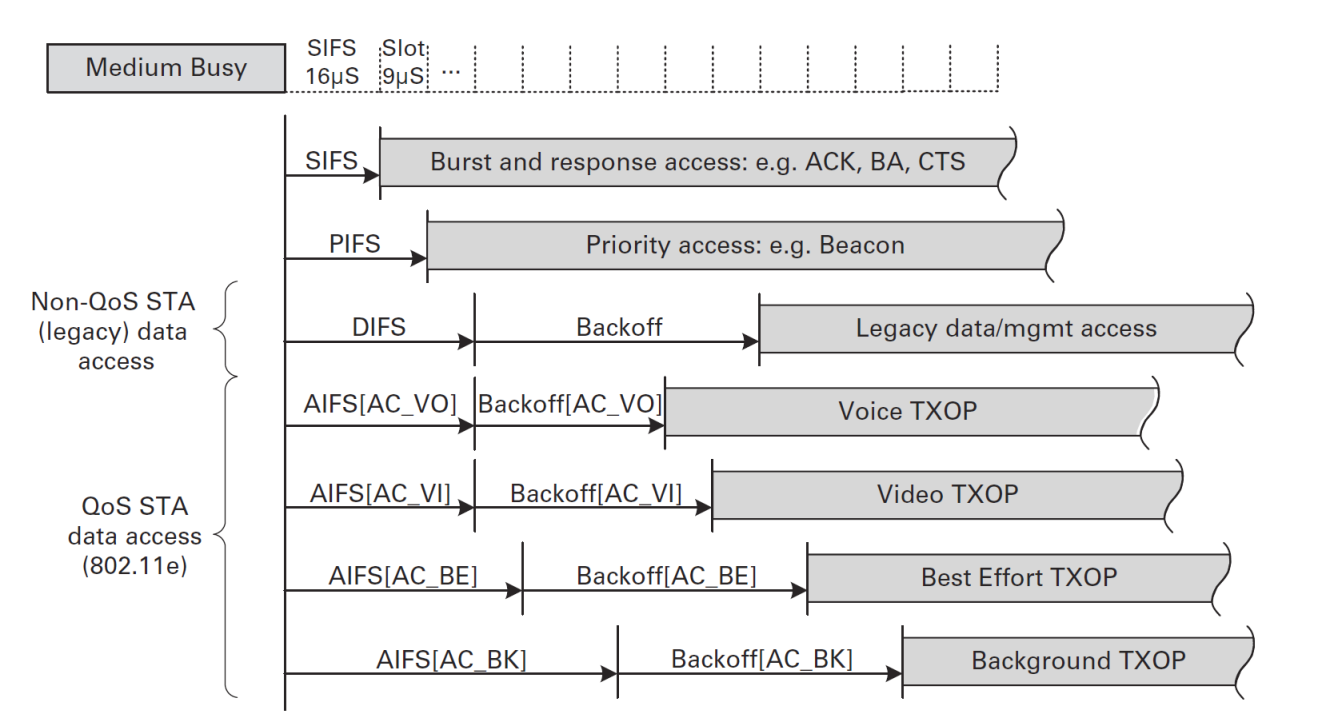


Figure 2 Access category illustration.

* IEEE 802.11系统是一个以分布式调度为主的系统。因此优先级对每个用户具有非常大的影响。从IEEE802.11e开始，引入了access category（接入类别）的概念。主要的category包括voice，video， best effort， background。其中voice的优先级别最高，video的优先级别次之，best effort的优先级别再次之，background的优先级别最低。

不同的access category用户采用不同的AIFS时隙，AIFS时隙越小，用户获取信道的可能性越大；AIFS时隙越大，用户获取信道的可能性越小。另一方面，不同的access category用户采用不同的contention window 大小。相类似，contention window 越小，用户获取信道的可能性越大；contention window越大，用户获取信道的可能性越小。

* SAiL：在可预见的SAiL 系统中，可以在一定程度上借鉴无线局域网中access category 的做法。比如LTE control signaling 可以获得更高的优先级别，而LTE data signaling 可以获得较低的优先级别。另外一个可能的例子是设计一种离散式的Proportional fairness的方案，使得每个用户的优先级（以及每个SAiL小区的优先级）取决于它的proportional fairness metric。不同于传统LTE系统中的proportional fairness，离散系统下的proportional fairness需要各个用户自己调整自己的优先级别。

### 空闲信道评估

* LTE是一个集中式调度的系统。因此不需要空闲信道评估。
* IEEE 802.11无线局域网中，空闲信道评估指clear channel assessment CCA，用于载波侦听。具体而言，CCA分为两种方式即静态CCA和动态CCA。静态CCA又分为基于信号能量的静态CCA模式和基于信号波形的静态CCA模式两种。这两种静态CCA模式在IEEE 802.11无线局域网中都被应用到。

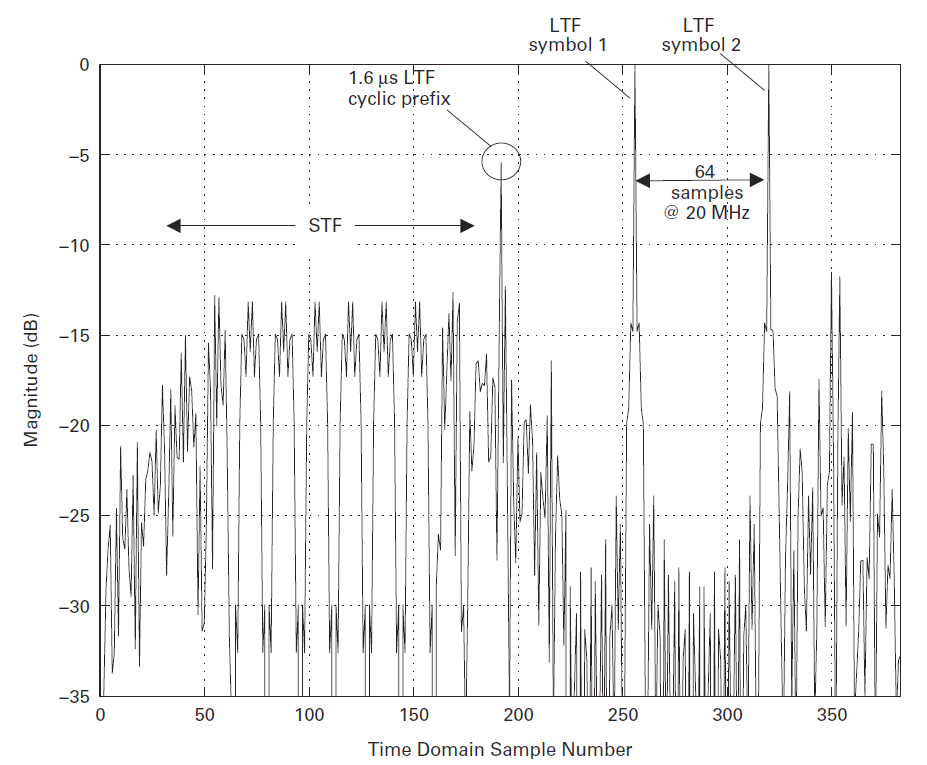


Figure 3 WLAN STF and LTF Illustration.

具体来说，在任何一个STA发射信号之前，STA都会测量其接收到的信号强度。如果接收信号强度超过某一个固定（静态）的门限值，则STA认为当前信道被其他STA占用，则进入backoff/defer状态。反之，如果接收信号强度低于某一个固定（静态）的门限值，则STA认为当前信道为空，则可以进入发射状态。这种模式可以称为基于信号能量的静态CCA模式。

另外，在任何一个STA发射信号之前，STA都会将接收到的信号波形与已知的STF（短训练序列）与LTF（长训练序列）做相关处理检测（可以为自相关或互相关）。如果相关检测结果为正，则STA认为当前信道被其他STA占用，则进入backoff/defer状态。反之，如果相关检测结果为负，则STA认为当前信道为空，则可以进入发射状态。这种模式可以称为基于信号波形的静态CCA模式。

动态空闲信道评估CCA与静态CCA基本类似。所不同的是门限值可变。如果CCA是基于信号能量，则判决信道是否被占用的门限值本身不是固定的，而是随接收机位置以及其他因素而变化。例如，对一个离AP较近的STA 来说，该STA可能进行信道接入的概率可以较高而不影响整个网络的吞吐量。在这种情况下，该STA的CCA门限可以设的较高（即该STA可以更加aggressive地获取信道）；相反对一个离AP较远的STA来说，该STA可能进行信道接入的概率不能较高，否则会影响这个网络的吞吐量。在这种情况下，该STA的CCA门限应该设的较低（即该STA需要更加conservative地获取信道）。相类似，如果CCA是基于信号波形，则判决信道是否被占用的门限值本身也可以是可调的。在此不再详细说明。

* SAiL：在可预见的SAiL 系统中，由于需要和其他无线局域网用户或其他SAiL用户共同竞争有限的unlicensed频率资源，因此，SAiL系统需要借鉴类似的CCA机制，或在此之上做进一步的演进。

### 许可辅助访问(LAA)

许可辅助访问（LAA）是LTE在没有执照的频谱中应用的有效方法，在许可频谱中携带LTE信号的两个分量载波同时使用，二级载体（无照频谱）以一种巧妙的方式应用。在这个情况下，主载波总是有效的，而二级（无证）载体的有效与否取决于信道。LTE中一个大的基站可能用于覆盖一个大的范围，和覆盖很多小的用户机，这些小的社区都配置了LAA接入点，如图4所示。

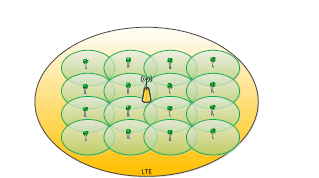


Figure 4 LTE macro cells and LAA small cells.

LBT是一项能与LAA现存的技术（比如WiFi）相兼容的重要技术:载波监听。简单的说就是一个LAA发射机确定他是否可以进行传输需要监听无线信道（在无执照频谱）。如果信道非常繁忙，那么发射机需要保持准备发送的状态。否则，任何现有的传输（例如，任何现有的或通过WiFi设备或者其他LAA设备）可能会严重干扰。如果信道是空闲的，那么发射机可能在未经授权的频谱上继续传输。相应的技术已经应用于IEEE802.11。

### 静态空闲信道评估

在IEEE802.11领域，载波监听的特点空闲信道评估（CCA）。空闲信道评估技术（CCA）的应用基于两种形式，其中一种是基于信号能量的CCA，一种是基于信号波形的CCA。由于基于信号波形的CCA和IEEE802.11的波形格式的联系非常的紧密，所以这里我们仅仅简单的考虑基于信号能量的CCA。基于信号能量的CCA:发射机需要检测在信号带宽内总的接受功率，在这里非相干操作不需要有任何发射机的信号结构或数据包格式的反馈。在IEEE802.11中，当在带宽20MHz下总的接受功率是一个大于62dBm，这种情况下视为信道繁忙，发射机应该申请信道。

CCA机制的本质是监听LAA发射机。当设置CCA的阈值是一个比较的小的值，发射机将会对现有的通信非常的敏感。甚至这个发射机会持续的向很远的地方发射信号。因此，对于任何的一个发射机是一个非常小的机会进行发射信号。另外的一个方面，如果设置CCA的阈值为一个很大的值，那么发射机对于发射自身数据过于积极，忽略了比较远的发射机并并且只关心比较近的发射机。这样的造成的结果就是任何发射机都有一个非常大的机会发射信号。静态CCA的值可能将LAA社区内所有的移动设备设置相同的值。上面所述就是已经被IEEE802.11采纳静态CCA策略。

### 动态空闲信道评估

另外一种可能的方式是根据接受信号的强度来设置CCA的值[9]。他是通过让移动站点（mobile station）监控一个周期或半个周期的从服务站点（serving mobile station）发送到移动站的信标。

移动站点（mobile station）从提供服务LAA基站获取一个功率高的参考信号，这样移动站点（mobile station）将有更多机会接触小区的用户，并且小概率造成对其他用户产生冲突，因此我们需要有一个高的CCA值，使其更加积极的（more aggressive）访问未经授权的频谱。在另外的一方面，移动站点（mobile station）从提供服务的LAA基站（甚至边缘用户）获取一个低功率的参考信号，移动站点（mobile station）有更多的机会接触小区的边缘用户，但是会造成和其他用户产生冲突的概率过高，因此需要有一个比较小的CCA值，并且更加保守（more conservative）的访问未经授权的频谱。以上就是动态CCA策略，并且IEEE802.11ax正在讨论标准化。

虽然静态CCA的优势是对所有的用户在相同的网络中都有相同的机会来访问信道，因此更加公平。但是如果给所有的远近设备设置静态CCA的值，对于边缘用户来说他们具有差的信道和高碰撞率，这样是不公平的，并且拉低了整个网络的吞吐量。

### 发射功率控制

每一个发射机（transmitter）使用特定的发射功率。在理想条件下，发射机（transmitter）的发射功率是根据发射机（transmitter）和接收机（receiver）之间的路径损耗来确定的，这就是所谓的动态发射功率。在实践中这种动态的发射功率设定是不可靠的，因为这样使得干扰水平在无执照频谱中会改变的频繁。信号的损失也是需要考虑的，特别是小数据包传输。通常大多数WiFi发射机是使用固定的发射功率。

我们在LAA系统中可以使用固定策略来控制发射功率，这就是发射功率控制。在LAA网络中设置发射功率控制是指占有无线介质的发射机以多大声音说。假设我们发射功率设置为比较小的值（比如10dBm），这样仅仅是相邻的用户才能监听并且持续的传输自己的数据，比较远的用户就不能监听到并且无法传输数据。这样就造成一部分的用户就有很大的机会去传输，另外一部分用户机会就很少。如果我们设置发射功率是一个很大的值（比如20dBm）,这样不仅临近的用户而且远的用户都能够监听到并且传输他们的数据。造成的结果是所有的用户通常都只有很小的概率进行传输。

正确设置CCA参数和小区内的发射功率是非常重要的。下面在LAA网络系统中进行系统的数值评估和学习CCA和TPC的交互。

### 评估方法

1. 模拟设定

为了简单起见，我们认为这里有16个房间，每个房间宽10米，长10米，高3米。这16个房间组成一个大的区域。如图1所示，其中每行4个房间，每列有4个房间。每个房间都有一个放置在房间的中心天花板上的LAA小基站。对于每一个房间，四个LAA移动设备随机的放置在房间里面，其高度是处于大于0米和小于2米的区间内。假设在相同的房间里每一个LAA基站服务每一个移动站点的机会都是一样的。对于所有的LAA基站我们使用的是载波频率是5.2GHz。除非明确的提到，频率复用因子3的含义是每个LAA基站上随机选择3个20MHz带宽进行操作。这个模型中的频率复用是托管的，在其他的案例中，频率复用可能将严格的管制，且受空间大小的限制使用相同频谱的可能性很大。

室内信道类型B（Indoor channel type B）使用的是在IEEE802.11ac的信道模型已经定义的信道模型，其中不仅包含路径损耗（path loss,），对数正态分布阴影衰落（log-normal shadow fading），还包含多径衰落（multipath channel fading）。这样的信道模型适合小的室内房间。模型中假设每个房间都是有墙壁，墙壁的渗透是有损失的，并且我们假定渗透一面墙壁的损耗为14dB。如果两个房间是处于对角线上，这里猜想他们是由两个墙壁阻挡，这样就造成28dB的渗透损耗。噪声功率的假设用房间的温度和5dB噪声系数（noise figure）。这里只考虑上行传输，即所有的数据流量都是移动站点（mobile stations）到LAA小区基站（LAA small cell base stations），并且我们 假设是一个全缓冲交互模式。这里规定所有的LAA移动站点（mobile stations）使用设置CCA参数和提供服务的基站（cell base stations.）设置TPC参数。

1. 评估策略

数值评估（numerical evaluation）是在多个场景（drop）下完成的，每一个场景（drop）下，移动站点（mobile stations）的位置是随机产生的并且保持不变，路径损耗（path loss）是通过传输者（干扰者）和接收者之间的距离并进行相应的计算，阴影衰减是随机产生的并且保持不变。对于每一个场景中我们有多个事件进行模拟，每一个事件对应一组链接组合（link）（每一个链接组合有一个发射机和一个接收机），在同一个时刻这一组链接组合（link）通过满足CCA参数约束和发射功率限制下进行有序通信。建模中使用的是随机信道访问的形式，使得链接组合（link）的集合因为不同的访问顺序而不同。对于每一个场景每一个事件，都是采取以下的CCA和TPC策略步骤访问信道：

1.设置初始化链接

随机选取一个房间，在房间内选择LAA基站和随机选择一个LAA移动站点。在上行传输，这个LAA移动站点是一个发射机（transmitter），而LAA基站是一个接收机（receiver）。在下行传输中，他们的角色是互换。对于每一个特定环节的链接组合（link），我们是根据信道计算路径损耗（path loss）和产生随机的阴影衰落（random shadow fading）和随机的多径衰落（random multipath fading）。最后我们标记这个房间被访问过。

2.增加新的链接

在剩下没有访问过的房间内随机选择一个房间。同样的，在这个房间内我们选择LAA基站和随机选择一个LAA移动站点。在上行传输中，这个LAA移动站点是一个发射机，LAA基站是一个接受机。在下行传输中，他们的角色互换。

在判断链接组合（link）是否激活之前，新发射机（transmitter）需要进行空闲信道评估（CCA）。同时新的发射机（transmitter）需要计算他所接受到来自其他已经激活的发射机（transmitter）干扰（潜在的干扰）的总功率。如果新的发射机（transmitter）接受到的功率大于CCA的阈值，那么这个新的连接就不能被激活，他们之间就不能进行数据传输。否者这个新的连接可以被激活，他们之间也可以进行通信，以此同时相关的发射机（transmitter）标记为激活。无论新的链接（link）是否被激活，最后这个房间标记都标记为被访问过。将上述步骤重复操作直到在这个系统中的所有的房间都被访问过。在这里需要注意的是在一个网络中有N个LAA基站，以此同时最多会有N个链接（link）被激活。

3.SINR和吞吐量计算

在这一步中，所有的房间被标记了，并且确定了激活的链接组合（links）数（相关的发射机和接收机）。对于每一个激活的接收机（receiver），我们需要计算SINR（信干噪比），SINR是考虑接受来自本链接发射机自身干扰信号和来自其他被激活的发射机产生的干扰信号。例如，有n个已经激活的链接，每一个接受者都对应有一个提供服务的链接和n-1个干扰链接。接受到的SINR映射到可以进行传输数据的链接中，SINR的映射是通过使用MCS（调制和解码）查找表。通常使用的产生MCS表是链接组合仿真出来的。对于每一个事件，通过所有激活链接来计算总的吞吐量。

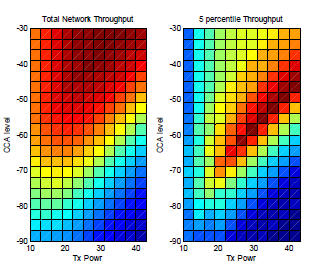


Figure 5 Static CCA and TPC.

而且在每一个场景下我们都重复多个事件，一些站点可能被激活（有吞吐量），在下一次事件中可能没被激活（没有吞吐量）。因此对于每一个站点，我们在所有的事件和所有的场景中就定义为每一个用户的平均吞吐量。这样的目的是通过观察每个用户的吞吐量来研究用户吞吐量的怎么分布的。重要的度量标准是观察5%用户吞吐量，在某种程度上反映了系统公平。

# 第二章仿真结果与分析

本章主要介绍了CCA以及TXP参数优化方案的仿真源代码以及仿真结果。本仿真主要是仿真上行网络。仿真的过程是基于静态txp频率复子采用1 和3 两种情况的，每个频率复用因子仿真五组，五组中参数设置如下表所示：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 组 号 | 第一组 | 第二组 | 第三组 | 第四组 | 第五组 |
| 描 述 | 静态CCA，新旧用户CCA值相同 | 动态CCA，新旧用户CCA Margin 相同 | 静态CCA，新旧用户CCA不同 | 动静结合 | 全动态CCA |
| cca Scheme | 0 | 1 | 0 | 2 | 1 |
| ccaLevel(dBm)或ccaMargin(dB)  (Old) | -90:5：-40 | 0:5:50 | -90:5：-40 | -90:5：-40 | 0:5：50 |
| ccaLevel(dBm)或ccaMargin(dB)  (New) | -90:5：-40 | 0:5:50 | -80:5：-30 | 0:5:50 | 0:5:50 |
| Txp Scheme | 0 | 0 | 0 | 0 | 0 |
| Txp Level(dBm)或  txp Margin(dB) | 3:3:40 | 3:3:40 | 3:3:40 | 20 | 20 |

仿真程序源代码

1. 参数设置

% frFactor, ... % 频率复用因子；如果为1，所有房间（或小区）公用同一频点，

% % 因此不同房间之间会相互干扰

% ccaScheme, …. % CCA方案

% ccaLevel, … % CCA参数设置；如果是静态CCA，则为CCA参数(dBm)；如果是

% % 动态CCA,则设置为CCA margin(dB)

% txpScheme, …. % 发送功率方案

% ccaLevel, … % 发送功率参数；如果是静态TXP，则为TXP参数(dBm)；如果是

% % 动态TXP,则设置为TXP margin(dB)

nets.frFactor = frFactor;

nets.ccaScheme = ccaScheme;

nets.ccaLevelOld = ccaLevelOld;

nets.ccaLevelNew = ccaLevelNew;

nets.txpScheme = txpScheme;

nets.txpLevelOld = txpLevelOld;

nets.txpLevelNew = txpLevelNew;

nets.mcstable = 1; % MCS参数设置

nets.fixAPTxpLevel = 20; % 对于上行信道所有服务中的AP传输功率默认20dBm

nets.fixCCALevel = -80; % CCA值默认为-80 dBm

nets.numDrops = 500; % 场景数

nets.numEvents = 20; % 每个场景下随机事件数

nets.roomSize = 10; % 房间长度 = 宽度

nets.roomHeight = 3; % 房间高度

nets.floorLoss = 17; % floor衰减

nets.wallLoss = 12; % reference IEEE 802.11-14/0082

nets.combLoss = 14.5; % combined floor/wall loss

ccaLevelAP = nets.fixCCALevel; % not used for uplink

ccaLevelOldSTA = nets.ccaLevelOld; % to be configured

ccaLevelNewSTA = nets.ccaLevelNew; % to be configured

txpLevelAP = nets.fixAPTxpLevel; % AP Txp 参数用来确定STA的 动态CCA参数

txpLevelOldSTA = nets.txpLevelOld; % to be configured

txpLevelNewSTA = txpLevelNew; % to be configured

uldlProb = 1;

1. 网络设置

在我们此次的仿真中，我们在每个房间安装一个AP，并在每个房间随机产生四个用户，其中有两个旧用户两个新用户。具体的仿真代码如下所示：

show3D = 0; % 显示；默认关闭此功能

numFloors = 1; % 整个网络的层数

numRows = 5; % 行数

numCells = 5; % 每行的房间/小区数

numOldSTAperAP = 4; % 每个AP/room中旧用户的数目

numNewSTAperAP = 4; % 每个AP/room中新用户的数目

numRooms = numFloors \* numRows \* numCells; % 宗房间数

numAP = numRooms; % 每个房间一个AP

numSTAsperAP = numOldSTAperAP + numNewSTAperAP; % 每个AP下的用户数

nodesperAP = numSTAsperAP + 1; % 每房间设备数

numSTAs = numSTAsperAP \* numAP; % 整个网络用户数

numOldSTAs = numOldSTAperAP \* numAP; % 整个网络旧用户数

numNewSTAs = numNewSTAperAP \* numAP; % 整个网络新用户数

numNodes = numAP + numSTAs; % 整个网络设备数

nets.show3D = show3D;

nets.floors = numFloors;

nets.rows = numRows;

nets.cells = numCells;

nets.numOldSTAperAP = numOldSTAperAP;

nets.numNewSTAperAP = numNewSTAperAP;

nets.numRooms = numRooms;

nets.numAP = numAP;

nets.numSTAsperAP = numSTAsperAP;

nets.nodesperAP = nodesperAP;

nets.numSTAs = numSTAs;

nets.numOldSTAs = numOldSTAs;

nets.numNewSTAs = numNewSTAs;

nets.numNodes = numNodes;

nets.freqIndex = 1;

nets.nodeIndex = 1:nets.numNodes; % 节点编号

nets.APIndex = 1:nets.nodesperAP:nets.numNodes; % AP编号

nets.STAIndex = setdiff(nets.nodeIndex, nets.APIndex); % 用户编号

oldSTAIndexTemp = [];

newSTAIndexTemp = [];

for n=1:1:nets.numOldSTAperAP

oldSTAIndexTemp = [oldSTAIndexTemp n+1:nets.nodesperAP:nets.numNodes];

newSTAIndexTemp =

[newSTAIndexTemp n+1+numOldSTAperAP:nets.nodesperAP:nets.numNodes];

end

nets.oldSTAIndex = sort(oldSTAIndexTemp);

nets.newSTAIndex = sort(newSTAIndexTemp);

nets.uldlProb = uldlProb;

1. 房间设置

本次仿真中我们假定每个房间的长度等于宽度，我们也可以不这样设置，在代码中将其设置成不同长度也是被允许的。而在每个房间中我们设置一个AP，其位于房间的正中间且位置固定。在每个房间中有四个随机产生的用户。

* 在第X层第Y行第Z列加入一个房间：

for x=1:1:numFloors

for y=1:numRows

for z=1:numCells

roomID = roomID+1; % 房间号

roomArray(roomID) = rooms(roomID,x,y,z,nets); % 在第X层第Y行第Z列加入一个房间

end;

end;

end

* 在第X层第Y行第Z列加入一个房间的具体实现：

function obj = rooms (roomid, floorno, rowno, cellno, nets)

obj.x\_length = nets.roomSize; % 房间在X轴上的长度

obj.y\_length = nets.roomSize; % 房间在Y轴上的长度

obj.z\_length = nets.roomHeight; % 房间在Z轴上的长度

obj.x\_start = (cellno - 1) \* obj.x\_length; % 房间在X轴上的起始坐标

obj.y\_start = (rowno - 1) \* (obj.y\_length + 0); % 房间在Y轴上的起始坐标

obj.z\_start = (floorno - 1) \* obj.z\_length; % 房间在Z轴上的起始坐标

obj.x\_end = obj.x\_start + nets.roomSize; % 房间在X轴上的终止坐标

obj.y\_end = obj.y\_start + nets.roomSize; % 房间在Y轴上的终止坐标

obj.z\_end = obj.z\_start + nets.roomHeight; % 房间在Z轴上的终止坐标

obj.x\_middle = mean([obj.x\_start obj.x\_end]); % 房间在X轴上的中间坐标

obj.y\_middle = mean([obj.y\_start obj.y\_end]); % 房间在Y轴上的中间坐标

obj.z\_middle = mean([obj.z\_start obj.z\_end]); % 房间在Z轴上的中间坐标

obj.roomid = roomid; % 房间号

obj.floorno = floorno; % 房间层号

obj.rowno = rowno; % 房间行号

obj.cellno = cellno; % 房间列号

obj.freq = nets.freqIndex; % 频率索引

end

* 新增一个AP

function ap = addAP(roomArray, roomid, nodeid, txpLevel, ccaLevel, freqindex, nets)

% 在房间的正中间安装一个AP

ap.x = roomArray(roomid).x\_middle;

ap.y = roomArray(roomid).y\_middle;

ap.z = roomArray(roomid).z\_start + 0.5 \* roomArray(roomid).z\_length;

ap.floorno = roomArray(roomid).floorno;

ap.rowno = roomArray(roomid).rowno;

ap.cellno = roomArray(roomid).cellno;

ap.roomid = roomid;

ap.nodeid = nodeid;

ap.txpLevel = txpLevel;

ap.ccaLevel = ccaLevel;

ap.freq = freqindex;

ap.old = 1;

if nets.show3D == 1 % 三维显示

plot3(ap.x,ap.y,ap.z,'k o');hold on;

end

end

* 新增一个STA

function sta = addSTA(roomArray, roomid, nodeid, txpLevel, ccaLevel, freqindex, nets, oldsta)

sta.x = roomArray(roomid).x\_start + rand \* nets.roomSize; % 随机产生STA的X坐标

sta.y = roomArray(roomid).y\_start + rand \* nets.roomSize; % 随机产生STA的Y坐标

sta.z = roomArray(roomid).z\_start + rand \* 2; % 随机产生STA的Z坐标

sta.floorno = roomArray(roomid).floorno;

sta.rowno = roomArray(roomid).rowno;

sta.cellno = roomArray(roomid).cellno;

sta.roomid = roomid;

sta.nodeid = nodeid;

sta.txpLevel = txpLevel;

sta.ccaLevel = ccaLevel;

sta.freq = freqindex;

if nets.show3D == 1 % 三维显示

if oldsta == 1

plot3(sta.x,sta.y,sta.z,'k .');hold on;

else

plot3(sta.x,sta.y,sta.z,'k .');hold on;

end

end

if oldsta == 1

sta.old = 1;

else

sta.old = 0;

end

end

1. 场景循环

在模拟的场景循环中，对于每个场景内，我们假设每个用户的位置、信道衰减以及信道阴影等保持不变。而在不同的场景之间，每个用户的位置、信道衰减以及信道阴影等则是随机变化的。

for xDrops = 1:nets.numDrops

freqArray = randi([1 frFactor],1,numRooms); % 给每个房间随机分配一个频点

nodeID = 0;

for roomID=1:numRooms % 对每个房间进行设备安装

nodeID = nodeID+1; % 当前设备(AP或STA)的nodeID

APID = nodeID; % AP的ID

% AP安装：房间号、设备号、发射功率、CCA参数、频点、nets系统参数

nodes(nodeID) = roomArray.addAP(roomID, nodeID, txpLevelAP, ccaLevelAP,freqArray(roomID), nets);

for j=1:nets.numOldSTAperAP

nodeID = nodeID+1; % 当前用户设备号nodeID

% 房间号，设备号，发射功率，CCA，频点，nets，old STA

nodes(nodeID) = roomArray.addSTA(roomID,nodeID,txpLevelOldSTA, ccaLevelOldSTA, freqArray(roomID), nets, 1);

ccaLevel = nets.ccaLevelOld;

txpLevel = nets.txpLevelOld;

if ccaScheme == 1 % 如果动态CCA且上行，需要调整每个用户的CCA参数

[pathLoss, ~, dist] = calcPL(nodes(APID), nodes(nodeID),nets); % 根据位置计算信道衰减

% 动态CCA=AP发射功率-信道衰减-CCA margin参数

ttcca = nets.fixAPTxpLevel - pathLoss - ccaLevel;

if ttcca > -50 % 单位dBm

ttcca = -50;

end

nodes(nodeID).ccaLevel = ttcca;

end

if txpScheme==1 % 如果动态功率控制且上行，调整发射功率

[pathLoss, ~, dist] = calcPL(nodes(APID), nodes(nodeID),nets);

targetrxpwr = 30 + (-96); % 期望接受功率；期望SNR设为30dBm。

% (-96)= (-174)+73(20MHz)+5(noise figure)

targettxpwr = targetrxpwr + pathLoss + txpLevel; % 期望发射功率

if targettxpwr < 0

targettxpwr = 0;

end

nodes(nodeID).txpLevel = targettxpwr;

end

end

nodes\_f1 = [];

nodes\_f2 = [];

nodes\_f3 = [];

nodes\_tt = [];

for nodeID=1:nets.numNodes

nodes\_tt = [nodes\_tt nodes(nodeID)]; % ËùÓÐÆµµãÉÏµÄÍøÂçÒÔ¼°Éè±¸ÁÐ±í

switch nodes(nodeID).freq

case 1

nodes\_f1 = [nodes\_f1 nodes(nodeID)]; % µÚÒ»ÆµµãÉÏµÄÍøÂçÒÔ¼°Éè±¸ÁÐ±í

case 2

nodes\_f2 = [nodes\_f2 nodes(nodeID)]; % µÚ¶þÆµµãÉÏµÄÍøÂçÒÔ¼°Éè±¸ÁÐ±í

case 3

nodes\_f3 = [nodes\_f3 nodes(nodeID)]; % µÚÈýÆµµãÉÏµÄÍøÂçÒÔ¼°Éè±¸ÁÐ±í

end

end

% 重复上面过程 对于新的STA

for j=1:numNewSTAperAP

nodeID = nodeID+1;

% 房间号，设备号，发射功率，CCA，频点，nets，new STA

nodes(nodeID) = roomArray.addSTA(roomID,nodeID,txpLevelNewSTA, ccaLevelNewSTA, freqArray(roomID), nets, 0);

ccaLevel = nets.ccaLevelNew;

txpLevel = nets.txpLevelNew;

if ccaScheme >0

[pathLoss, ~, dist] = calcPL(nodes(APID), nodes(nodeID),nets);

ttcca = nets.fixAPTxpLevel - pathLoss - ccaLevel;

if ttcca > -50

ttcca = -50;

end

nodes(nodeID).ccaLevel = ttcca;

end

if txpScheme >0

[pathLoss, ~, dist] = calcPL(nodes(APID), nodes(nodeID),nets);

targetrxpwr = 30 + (-96); % (-96) = (-174) +73(20MHz) + 5(noise figure);

targettxpwr = targetrxpwr + pathLoss + txpLevel; % txpLevel is the margin here.

if targettxpwr < 0 % 单位dBm

targettxpwr = 0;

end

nodes(nodeID).txpLevel = targettxpwr;

end

end

end

nodes\_f1 = [];

nodes\_f2 = [];

nodes\_f3 = [];

nodes\_tt = [];

for nodeID=1:nets.numNodes

nodes\_tt = [nodes\_tt nodes(nodeID)]; % 所有频点上的网络以及设备列表

switch nodes(nodeID).freq

case 1

nodes\_f1 = [nodes\_f1 nodes(nodeID)]; % 第一频点上的网络以及设备列表

case 2

nodes\_f2 = [nodes\_f2 nodes(nodeID)]; % 第二频点上的网络以及设备列表

case 3

nodes\_f3 = [nodes\_f3 nodes(nodeID)]; % 第三频点上的网络以及设备列表

end

end

%% 仿真

% 本仿真中函数sim\_one\_freq(nodes\_f0, nodes\_tt, nets)函数为单独一个频点仿真的函数

% nodes\_f0 : 本频点网络设备列表

% nodes\_tt : 所有网络设备列表

% nets : 系统参数

% 函数返回值 : 返回当前网络的throughput矩阵

switch frFactor

case 1

cout{1}{xDrops} = sim\_one\_freq(nodes\_f1, nodes\_tt, nets);

for m=1:length(cout{1}{xDrops})

dout{xDrops}(m).ulRoomThru = cout{1}{xDrops}(m).ulRoomThru;

dout{xDrops}(m).ulUserThru = cout{1}{xDrops}(m).ulUserThru;

end

case 3

cout{1}{xDrops} = sim\_one\_freq(nodes\_f1, nodes\_tt, nets);

cout{2}{xDrops} = sim\_one\_freq(nodes\_f2, nodes\_tt, nets);

cout{3}{xDrops} = sim\_one\_freq(nodes\_f3, nodes\_tt, nets);

for m=1:length(cout{1}{xDrops})

dout{xDrops}(m).ulRoomThru = cout{1}{xDrops}(m).ulRoomThru + ...

cout{2}{xDrops}(m).ulRoomThru + ...

cout{3}{xDrops}(m).ulRoomThru;

dout{xDrops}(m).ulUserThru = cout{1}{xDrops}(m).ulUserThru + ...

cout{2}{xDrops}(m).ulUserThru + ...

cout{3}{xDrops}(m).ulUserThru;

end

end

1. 数据采集

在数据采集模块中我们进行了一系列的模拟环境的仿真，其中包括有下列数据采集的函数：

|  |  |
| --- | --- |
| 函 数 | 说 明 |
| sim\_one\_freq(nodes\_f0,nodes\_tt,nets) | 单独一个频点的仿真 |
| collect(dout, nets) | 采集数据 |
| runbss(links, pathlossdb, chpwrdb, nodes\_tt, nets) | 仿真当前网络的throughput |
| check\_link(oldlinks, newlink, pathlossdb, chpwrdb, nodes\_tt) | 判断新的连接是否满足CCA准则 |
| calcPL(node1, node2, nets) | 计算连接的信道衰减和信道系数 |
| readChanProfile(chtype, distance\_Tx\_Rx\_m) | 读取信道类型以及信道类型参数 |
| phy\_abs(sinrdb, nets) | 从信干噪比查询可达的吞吐量 |

* 单独一个频点的仿真

function tout = sim\_one\_freq(nodes\_f0, nodes\_tt, nets)

if isempty(nodes\_f0) % 本频点没有设备则直接清零退出

for xEvent = 1:nets.numEvents

tout(xEvent).ulRoomThru = zeros(1, nets.numRooms);

tout(xEvent).ulUserThru = zeros(1, nets.numNodes);

end

else

numRoomsTtl = nets.rows \* nets.cells \* nets.floors; % 总共房间数(此频点及其他频点)

numRoomsThisFreq = length(nodes\_f0)/nets.nodesperAP; % 此频点房间数

numNodesTtl = numRoomsTtl \* nets.nodesperAP; % 总共设备数

numNodesThisFreq = length(nodes\_f0); % 此频点设备数

numStasTtl = numRoomsTtl \* (nets.nodesperAP - 1); % 总共用户数

numStasThisFreq = numRoomsThisFreq \* (nets.nodesperAP - 1);% 此频点用户数

roomIdThisFreq = [];

nodeIdThisFreq = [];

for i = 1:length(nodes\_f0)

roomIdThisFreq = [roomIdThisFreq, nodes\_f0(i).roomid];

nodeIdThisFreq = [nodeIdThisFreq, nodes\_f0(i).nodeid]; % 此频点上的设备号列表

end

roomIdThisFreq = unique(roomIdThisFreq); % 此频点上的房间号列表

% 对此场景下，由于用户位置已固定，AP位置已固定，信道衰减因此确定。信道阴影也假设

% 为确定并作为信道衰减的一部分。

for i= 1:numNodesTtl

for j=i:numNodesTtl

[pathlossdb(i,j), shadowingdb(i,j), chpwrdb(:,i,j)] = calcPL(nodes\_tt(i), nodes\_tt(j), nets);

pathlossdb(i,j) = pathlossdb(i,j) + shadowingdb(i,j);

end

end

for i=1:numNodesTtl

for j=1:i

pathlossdb(i,j) = pathlossdb(j,i);

chpwrdb(:,i,j) = chpwrdb(:,j,i);

end

end

roomThru = zeros(nets.numEvents, numRoomsTtl); % 每个房间throughput列表

userThru = zeros(nets.numEvents, numStasTtl); % 每个用户throughput列表

% ÊÂ¼þÑ­»·£»²»Í¬ÊÂ¼þÖÐ£¬²»Í¬µÄ·¿¼äÖÐµÄ²»Í¬ÓÃ»§µÃµ½²»Í¬µÄthroughput£»

for xEvent = 1:nets.numEvents

oldlinks = []; % ÒÑÈ·ÈÏÁ¬½Ó£¨Âú×ãCCA×¼Ôò£©

newlink = []; % ´ýÈ·ÈÏÁ¬½Ó£¨¸ù¾ÝCCA×¼Ôò½øÐÐÈ·ÈÏ£©

pendingset = roomIdThisFreq; % ±¾ÆµµãÉÏ´ýÈ·ÈÏ·¿¼äÁÐ±í

pendingaps = length(pendingset); % ±¾ÆµµãÉÏ´ýÈ·ÈÏ·¿¼äÊý

while ~isempty(pendingset)

ftt = randi(pendingaps, 1, 1); % ´Ó´ýÈ·ÈÏ·¿¼äÖÐËæ»úÑ¡È¡

roomx = pendingset(ftt); % ±»Ñ¡ÖÐ·¿¼äµÄÈ«¾Ö·¿¼äºÅ(ËùÓÐÆµµã)

roomap = (roomx-1) \* nets.nodesperAP + 1; % ±»Ñ¡ÖÐ·¿¼äAPÖ®Éè±¸ºÅ

roomsta = roomap + randi(nets.nodesperAP-1,1,1); % ÔÚ±»Ñ¡ÖÐ·¿¼äÖÐËæ»úÑ¡È¡Ò»¸öÓÃ»§Ö®Éè±¸ºÅ

pendingset = pendingset([1:ftt-1 ftt+1:end]); % ¸üÐÂ

pendingaps = length(pendingset);

% ÉÏÐÐ

txNodeID = roomsta; % ·¢Éä»úÉè±¸ºÅ

rxNodeID = roomap; % ½ÓÊÕ»úÉè±¸ºÅ

uldl = 1;

newlink.txid = txNodeID; % ÐÂÁ¬½Ó·¢Éä»úÉè±¸ºÅ

newlink.rxid = rxNodeID; % ÐÂÁ¬½Ó½ÓÊÕ»úÉè±¸ºÅ

newlink.uldl = uldl;

oldlinks = check\_link(oldlinks, newlink, pathlossdb, chpwrdb, nodes\_tt); % È·ÈÏ¸ÃÁ¬½ÓÊÇ·ñÂú×ãCCA×¼Ôò

end

all\_active\_links = oldlinks; % ±¾³¡¾°±¾ÊÂ¼þÏÂ£¬ËùÓÐactiveµÄÁ¬½Ó¡£

tout(xEvent) = runbss(all\_active\_links, pathlossdb, chpwrdb, nodes\_tt, nets);

end

end

end

% 事件循环；不同事件中，不同的房间中的不同用户得到不同的throughput；

for xEvent = 1:nets.numEvents

oldlinks = []; % 已确认连接(满足CCA准则)

newlink = []; % 待确认连接(根据CCA准则进行确认)

pendingset = roomIdThisFreq; % 本频点上待确认房间列表

pendingaps = length(pendingset); % 本频点上待确认房间数

while ~isempty(pendingset)

ftt = randi(pendingaps, 1, 1); % 从待确认房间中随机选取

roomx = pendingset(ftt); % 被选中房间的全局房间号(所有频点)

roomap = (roomx-1) \* nets.nodesperAP + 1; % 被选中房间AP的设备号

roomsta = roomap + randi(nets.nodesperAP-1,1,1); % 在被选中房间中随机选取一个用户的

% 设备号

pendingset = pendingset([1:ftt-1 ftt+1:end]); % 更新

pendingaps = length(pendingset);

% 上行

txNodeID = roomsta; % 发射机设备号

rxNodeID = roomap; % 接收机设备号

uldl = 1;

newlink.txid = txNodeID; % 新连接发射机设备号

newlink.rxid = rxNodeID; % 新连接接收机设备号

newlink.uldl = uldl;

% 确认该连接是否满足CCA准则

oldlinks = check\_link(oldlinks, newlink, pathlossdb, chpwrdb, nodes\_tt);

end

all\_active\_links = oldlinks; % 本场景本事件下，所有active的连接

tout(xEvent) = runbss(all\_active\_links, pathlossdb, chpwrdb, nodes\_tt, nets);

end

end

end

* 采集数据

function cstat = collect(dout, nets)

tmp0 = zeros(nets.numDrops, nets.numEvents, nets.numRooms);

tmp1 = zeros(nets.numDrops, nets.numEvents);

for xDrops = 1:length(dout)

for xEvent = 1:length(dout{1})

tmp0(xDrops, xEvent, :) = dout{xDrops}(xEvent).ulRoomThru;

tmp1(xDrops, xEvent) = length(find(dout{xDrops}(xEvent).ulRoomThru));

end

end

cstat.ulRoomThru = squeeze(mean(mean(tmp0,1),2))'; % 每个房间throughput列表

cstat.ulRoomActv = mean(mean(tmp1)); % Active的房间数

cstat.ulRoomMean = mean(cstat.ulRoomThru); % 每个房间平均throughput

tmp00 = zeros(nets.numDrops, nets.numEvents, nets.numSTAs);

tmp10 = zeros(nets.numDrops, nets.numEvents, nets.numOldSTAs);

tmp20 = zeros(nets.numDrops, nets.numEvents, nets.numNewSTAs);

tmp11 = zeros(nets.numDrops, nets.numEvents);

tmp21 = zeros(nets.numDrops, nets.numEvents);

for xDrops = 1:length(dout)

for xEvent = 1:length(dout{1})

tmp00(xDrops, xEvent, :) = dout{xDrops}(xEvent).ulUserThru(nets.STAIndex);

tmp10(xDrops, xEvent, :) = dout{xDrops}(xEvent).ulUserThru(nets.oldSTAIndex);

tmp20(xDrops, xEvent, :) = dout{xDrops}(xEvent).ulUserThru(nets.newSTAIndex);

tmp01(xDrops, xEvent) = length(find(dout{xDrops}(xEvent).ulUserThru(nets.STAIndex)));

tmp11(xDrops, xEvent) = length(find(dout{xDrops}(xEvent).ulUserThru(nets.oldSTAIndex)));

tmp21(xDrops, xEvent) = length(find(dout{xDrops}(xEvent).ulUserThru(nets.newSTAIndex)));

end

end

cstat.ulUserThru = squeeze(mean(mean(tmp00,1),2))'; % 每个用户throughput列表

cstat.ulOldUserThru = squeeze(mean(mean(tmp10,1),2))'; % 每个新用户throughput列表

cstat.ulNewUserThru = squeeze(mean(mean(tmp20,1),2))'; % 每个旧用户throughput列表

cstat.ulUserActv = mean(mean(tmp01)); % Active的用户数

cstat.ulOldUserActv = mean(mean(tmp11)); % Active的旧用户数

cstat.ulNewUserActv = mean(mean(tmp21)); % Active的新用户数

cstat.ulUserMean = mean(cstat.ulUserThru); % 每个用户平均throughput

cstat.ulOldUserMean = mean(cstat.ulOldUserThru); % 每个旧用户throughput

cstat.ulNewUserMean = mean(cstat.ulNewUserThru); % 每个新用户throughput

end

仿真当前网络的throughput

function out = runbss(links, pathlossdb, chpwrdb, nodes\_tt, nets)

numRooms = nets.floors \* nets.rows \* nets.cells;

ulRoomThru = zeros(1, numRooms); % 上行每个房间throughput

ulUserThru = zeros(1, numRooms \* (nets.numSTAsperAP+1)); % 上行每个用户throughput

numlinks = length(links);

if numlinks == 1

txid = links(1).txid;

rxid = links(1).rxid;

signal\_pwr = chpwrdb(:,txid, rxid) - pathlossdb(txid,rxid) + nodes\_tt(txid).txpLevel;

interf\_pwr = -96; % -174+73(20MHz)+5(noise figure)

sinrdb = signal\_pwr - interf\_pwr;

thruputs = phy\_abs(sinrdb, nets); % 单位是Mbps

% 上行

staid = txid;

apid = rxid;

roomid = (apid-1)/(nets.numSTAsperAP+1)+1; % 从apid获取roodid

ulRoomThru(roomid) = thruputs;

ulUserThru(staid) = thruputs;

else

for i=1:numlinks

rxid = links(i).rxid;

nilink = 0; % 干扰连接数

for j=1:numlinks

txid = links(j).txid;

if j==i

sgpwr = chpwrdb(:, txid, rxid);

sgpls = pathlossdb(txid,rxid);

sgpwr = sgpwr - sgpls + nodes\_tt(txid).txpLevel;

else

nilink = nilink + 1;

inpwr(:, nilink) = chpwrdb(:, txid, rxid);

inpls(nilink) = pathlossdb(txid, rxid);

inpwr(:, nilink) = inpwr(:, nilink) - inpls(nilink) + nodes\_tt(txid).txpLevel;

end

end

inpwr = [inpwr -96\*ones(size(inpwr,1), 1)]; % add noise power of -96dbm;

signal\_pwr = sgpwr;

interf\_pwr = 10\*log10(sum(10.^(inpwr/10),2));

sinrdb = signal\_pwr - interf\_pwr;

thruputs(i) = phy\_abs(sinrdb, nets);

% 上行

staid = links(i).txid;

apid = links(i).rxid;

roomid = (apid-1)/(nets.numSTAsperAP+1)+1;

% 上行

ulRoomThru(roomid) = thruputs(i);

ulUserThru(staid) = thruputs(i);

end

end

out.ulRoomThru = ulRoomThru; % 上行，每个房间throughput

out.ulUserThru = ulUserThru; % 上行，每个用户throughput

end

* 判断新的连接是否满足CCA准则

function oldlinks = check\_link(oldlinks, newlink, pathlossdb, chpwrdb, nodes\_tt)

if isempty(oldlinks) % 如果之前此频点上没有任何连接，则新的连接一定可以被加入。 linkid = 1;

oldlinks(linkid).txid = newlink.txid;

oldlinks(linkid).rxid = newlink.rxid;

oldlinks(linkid).uldl = newlink.uldl; % 退出

else

% 确认方法：针对每个已确认连接的发射机，计算其在新连接发射机处的接收功率。

% 如果总接收功率超过CCA门限，则此新连接不可接入。

% 如果总接收功率低于CCA门限，则此新连接可以接入。

NumLinks = length(oldlinks); % 已确认连接数

rxttl = -inf; % 总接收功率

for i=1:NumLinks

txid = oldlinks(i).txid; % 已确认连接发射机的设备号

rxid = newlink.txid; % 新连接发射机的设备号（注意此处非新连接接收机） txpLevel(i) = nodes\_tt(txid).txpLevel; % 发射功率

totalloss(i) = pathlossdb(txid, rxid) +... % 总衰减（信道衰减+信道系数）

10\*log10(sum(10.^(chpwrdb(:,txid,rxid)/10)));

rxpwr(i) = txpLevel(i) - totalloss(i); % 此（连接发射机）贡献的接收功率

rxttl = 10\*log10(10^(rxttl/10) + 10^(rxpwr(i)/10)); % 总接收功率

end

if rxttl < nodes\_tt(rxid).ccaLevel % 可以接入;更新

oldlinks(NumLinks+1).txid = newlink.txid;

oldlinks(NumLinks+1).rxid = newlink.rxid;

oldlinks(NumLinks+1).uldl = newlink.uldl;

else %不可接入；不需更新

end

end

end

* 计算两个给定节点的信道衰落和信道系数

function [PathLoss\_dB, Shadowing\_dB, chpwrdb] = calcPL(node1, node2, nets)

floor1 = node1.floorno;

floor2 = node2.floorno;

row1 = node1.rowno;

row2 = node2.rowno;

cell1 = node1.cellno;

cell2 = node2.cellno;

floors = sort([floor1 floor2]);

rows = sort([row1 row2]);

cells = sort([cell1 cell2]);

floordiff = abs(floors(2)-floors(1)); % 层数差别

rowdiff = abs(rows(2)-rows(1)); % 行数差别

celldiff = abs(cells(2)-cells(1)); % 列数差别

walldiff = rowdiff + celldiff; % 间隔的墙数目

floordiff = floordiff^((floordiff+2)/(floordiff+1)-0.46);

walldiff = walldiff^((walldiff+2)/(walldiff+1)-0.46);

combdiff = (floordiff+walldiff)^( (floordiff+walldiff+2)/(floordiff+walldiff+1) - 0.46);

penetrationloss = nets.floorLoss \* floordiff + walldiff \* nets.wallLoss;

distance = sqrt((node1.x-node2.x)^2+(node1.y-node2.y)^2+(node1.z-node2.z)^2);

chtype = 'B';

CarrierFrequency\_Hz = 5.2e9;

[PDP\_dB, d\_BP\_m, K\_factor\_dB, ShadowFading\_std\_dB] = readChanProfile(chtype, distance);

if (distance > d\_BP\_m)

PathLoss\_dB = 20\*log10(4\*pi\*CarrierFrequency\_Hz/3e8) + 20\*log10(d\_BP\_m) + (35\*log10(distance/d\_BP\_m));

else

PathLoss\_dB = 20\*log10(4\*pi\*CarrierFrequency\_Hz/3e8) + 20\*log10(distance);

end;

Shadowing\_dB = randn \* ShadowFading\_std\_dB; % 信道阴影

PathLoss\_dB = PathLoss\_dB + penetrationloss; % 信道衰减含穿透效应

K = 10.^(K\_factor\_dB/10);

L = size(PDP\_dB,2);

PDP = 10.^(PDP\_dB(1,:)/10);

PDPsum = sum(PDP);

PDP = sqrt(PDP/PDPsum);

h = (randn(1,L) + 1j\*randn(1,L))/sqrt(2);

th = sqrt(K./(K+1)) + sqrt(1./(K+1)) .\* h .\*PDP; % 时间域

fh = fft(th, 64)/sqrt(64);

fh = fh';

chpwrdb = 10\*log10(abs(fh).^2); % 信道系数(绝对值平方 dB)

end

* 读取信道类型以及信道类型参数

function [PDP\_dB, d\_BP\_m, K\_factor\_dB, ShadowFading\_std\_dB] = readChanProfile(chtype, distance\_Tx\_Rx\_m)

switch chtype

case 'A'

PDP\_dB = [ 0;0]; % Average power [dB] and Relative delay (ns)

d\_BP\_m = 5;

if (distance\_Tx\_Rx\_m < d\_BP\_m)

K\_factor\_dB = [0,(-100).\*ones(1, size(AoA\_Rx\_deg, 2)-1)];

ShadowFading\_std\_dB = 3;

else

K\_factor\_dB = (-100).\*ones(1, size(AoA\_Rx\_deg, 2));

ShadowFading\_std\_dB = 4;

end;

case 'B'

PDP\_dB = [ 0 -5.4287 -2.5162 -5.8905 -9.1603 -12.5105 -15.6126 -18.7147 -21.8168; ...

0 10e-9 20e-9 30e-9 40e-9 50e-9 60e-9 70e-9 80e-9 ];

d\_BP\_m = 5;

if (distance\_Tx\_Rx\_m < d\_BP\_m)

K\_factor\_dB = [0,(-100).\*ones(1, size(PDP\_dB, 2)-1)];

ShadowFading\_std\_dB = 3;

else

K\_factor\_dB = (-100).\*ones(1, size(PDP\_dB, 2));

ShadowFading\_std\_dB = 4;

end;

case 'C'

PDP\_dB = [ 0 -2.1715 -4.3429 -6.5144 -8.6859 -10.8574 -4.3899 -6.5614 -8.7329 -10.9043 -13.7147 -15.8862 -18.0577 -20.2291; ...

0 10e-9 20e-9 30e-9 40e-9 50e-9 60e-9 70e-9 80e-9 90e-9 110e-9 140e-9 170e-9 200e-9]; % Relative delay (ns)

d\_BP\_m = 5;

if (distance\_Tx\_Rx\_m < d\_BP\_m)

K\_factor\_dB = [0,(-100).\*ones(1, size(PDP\_dB, 2)-1)];

ShadowFading\_std\_dB = 3;

else

K\_factor\_dB = (-100).\*ones(1, size(PDP\_dB, 2));

ShadowFading\_std\_dB = 5;

end;

case 'D'

PDP\_dB = [ 0 -0.9 -1.7 -2.6 -3.5 -4.3 -5.2 -6.1 -6.9 -7.8 -4.7 -7.3 -9.9 -12.5 -13.7 -18 -22.4 -26.7; ...

0 10e-9 20e-9 30e-9 40e-9 50e-9 60e-9 70e-9 80e-9 90e-9 110e-9 140e-9 170e-9 200e-9 240e-9 290e-9 340e-9 390e-9]; % Relative delay (ns)

d\_BP\_m = 10;

if (distance\_Tx\_Rx\_m < d\_BP\_m)

K\_factor\_dB = [3,(-100).\*ones(1, size(PDP\_dB, 2)-1)];

ShadowFading\_std\_dB = 3;

else

K\_factor\_dB = (-100).\*ones(1, size(PDP\_dB, 2));

ShadowFading\_std\_dB = 5;

end

case 'E'

PDP\_dB = [ -2.5 -3.0 -3.5 -3.9 0 -1.3 -2.6 -3.9 -3.4 -5.6 -7.7 -9.9 -12.1 -14.3 -15.4 -18.4 -20.7 -24.6; ...

0 10e-9 20e-9 30e-9 50e-9 80e-9 110e-9 140e-9 180e-9 230e-9 280e-9 330e-9 380e-9 430e-9 490e-9 560e-9 640e-9 730e-9]; % Relative delay (ns)

d\_BP\_m = 20;

if (distance\_Tx\_Rx\_m < d\_BP\_m)

K\_factor\_dB = [6,(-100).\*ones(1, size(PDP\_dB, 2)-1)];

ShadowFading\_std\_dB = 3;

else

K\_factor\_dB = (-100).\*ones(1, size(PDP\_dB, 2));

ShadowFading\_std\_dB = 6;

end;

case 'F'

PDP\_dB = [ -3.3 -3.6 -3.9 -4.2 0 -0.9 -1.7 -2.6 -1.5 -3.0 -4.4 -5.9 -5.3 -7.9 -9.4 -13.2 -16.3 -21.2; ...

0 10e-9 20e-9 30e-9 50e-9 80e-9 110e-9 140e-9 180e-9 230e-9 280e-9 330e-9 400e-9 490e-9 600e-9 730e-9 880e-9 1050e-9]; % Relative delay (ns)

d\_BP\_m = 30;

if (distance\_Tx\_Rx\_m < d\_BP\_m)

K\_factor\_dB = [6,(-100).\*ones(1, size(PDP\_dB, 2)-1)];

ShadowFading\_std\_dB = 3;

else

K\_factor\_dB = (-100).\*ones(1, size(PDP\_dB, 2));

ShadowFading\_std\_dB = 6;

end;

end;

end

* 从信干噪比查询可达的吞吐量

function thruputs = phy\_abs(sinrdb, nets)

sinr = 10.^(sinrdb/10);

thruput = mean(real(log2(1+sinr)));

sinreff = 2^thruput - 1;

sinreffdb = 10\*log10(sinreff);

switch nets.mcstable

case 1

sinrthr = [-inf 3.2 7.0 9.6 11.8 15.5 18.8 20.4 21.9 25.0 27.0];

mcsrate = [0 6.5 13.0 19.5 26.0 39.0 52.0 58.5 65.0 78.0 86.7];

case 2

sinrthr = [-inf 3.2 7.0 9.6 11.8 15.5 18.8 20.4 21.9 25.0 27.0];

mcsrate = [0 6.5 13.0 19.5 26.0 39.0 52.0 58.5 65.0 78.0 86.7];

end

x = find((sinreffdb-sinrthr>0),1,'last');

thruputs = mcsrate(x);

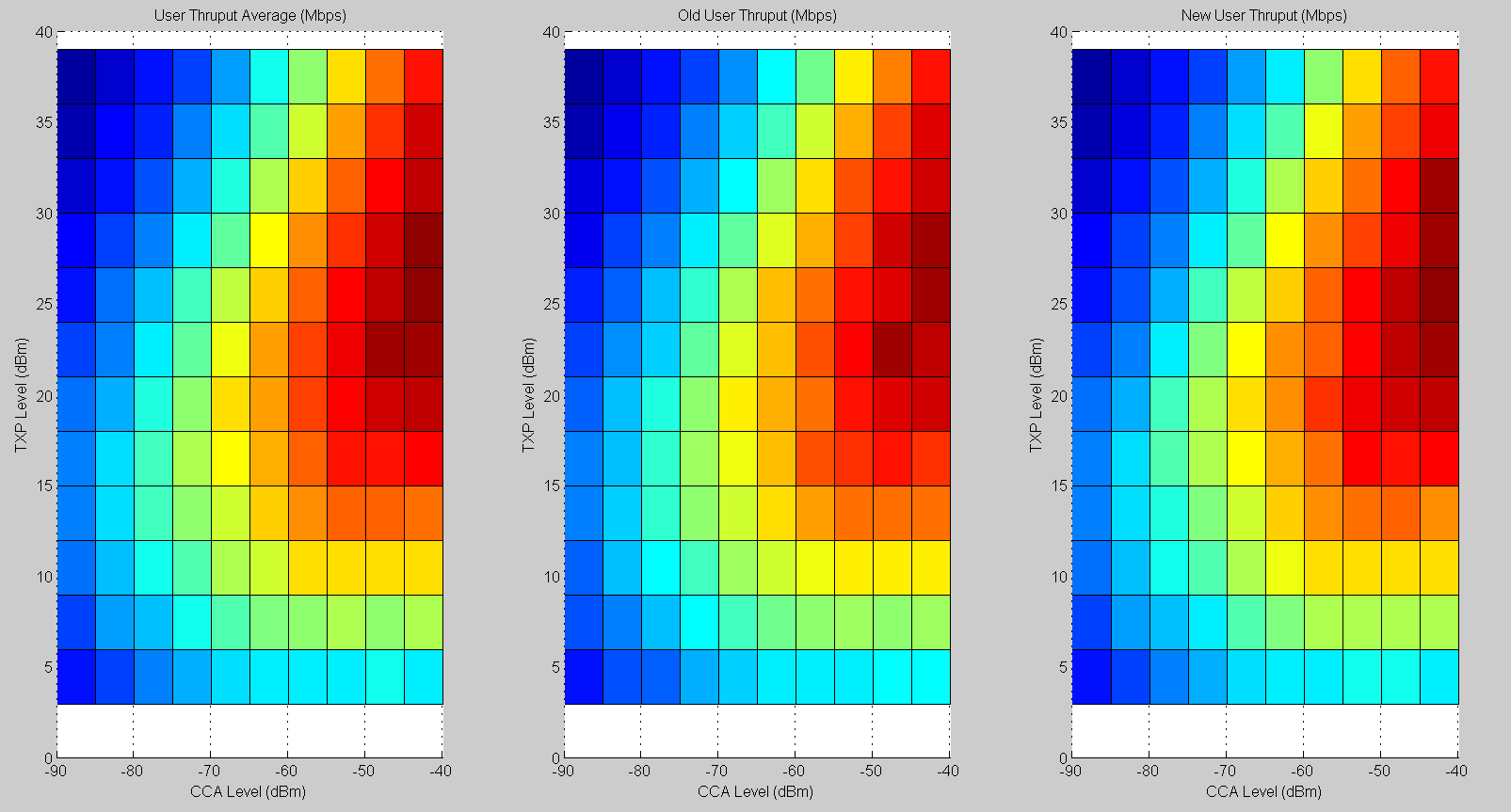
end

### 仿真结果（频率复用因子为1）

仿真图说明：

在下文所述的所有仿真图中，自左至右分别表示的是整个网络的平均吞吐量（User Thruput Average），旧用户的吞吐量(Old User Thruput)，新用户的吞吐量(New User Thruput)。每张图中颜色的深浅代表吞吐量的高低，颜色越深说明吞吐量越大，颜色越浅说明吞吐量就越小，而图中具体的横坐标以及纵坐标的含义以及横纵坐标的取值范围将会在每张仿真图的分析中进行介绍。下面将进行五组仿真结果的具体分析：

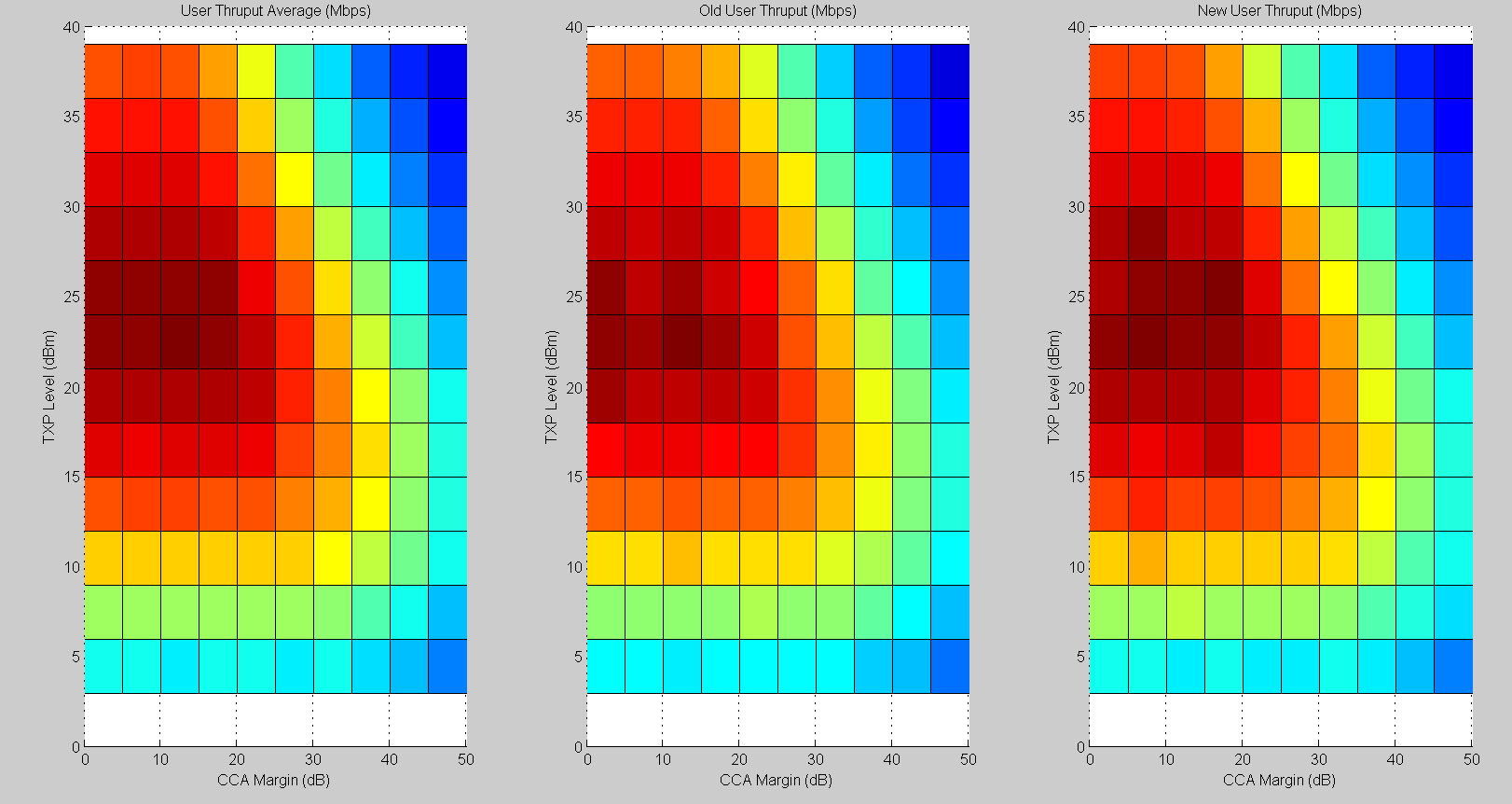
第一组：本组为纯静态CCA方案，新用户的CCA参数值与旧用户的参数值设置相同，新旧用户的CCA参数值均是从-90以5dBm为步长增加 到-40 dBm(横坐标),每个用户的TXP Level则是从3以3dBm为步长增加到40dBm（纵坐标）。具体的仿真结果如下图所示：



仿真结果分析：

对上图进行分析可以发现：无论是整个网络还是旧用户和新用户的吞吐量，其性能基本上都是一致的，可以表述为：随着CCA Level的增加，整个网络、旧用户以及新用户的吞吐量一开始都是增加随后达到饱和的；而随着TXP Level的增加，整个网络、旧用户以及新用户的吞吐量也是先增加随后达到饱和的。

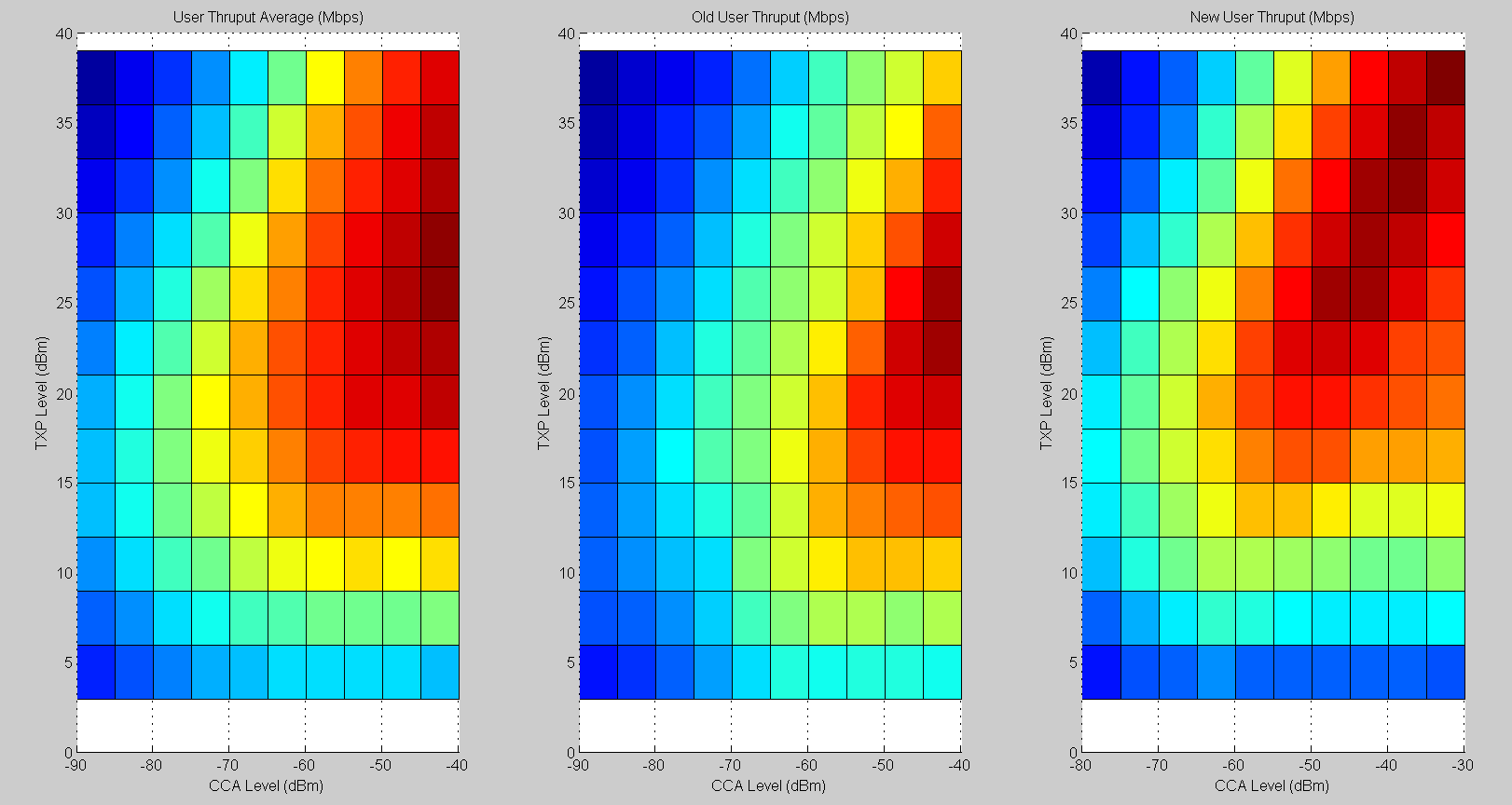
第二组：本组为纯动态CCA方案，新用户的CCA参数值与旧用户的CCA参数值设置相同，新旧用户的CCA Margin值均是从0 dB开始以5dB为步长增加到50 dB(横坐标),每个用户的TXP则是从3dBm开始以3dBm为步长增加到40dBm(纵坐标)。具体的仿真结果如下图所示：



仿真结果分析：

对上图进行分析可以发现：无论是整个网络还是旧用户和新用户的吞吐量，其性能基本上都是一致的，可以表述为：随着CCA Margin的增加，整个网络、旧用户以及新用户的吞吐量一开始都是慢慢降低，当降低到一定的程度的时候将不再降低；而随着TXP Level的增加，整个网络、旧用户以及新用户的吞吐量则是先增加到一定的值，而当TXP Level的值再继续增加的时候，整个网络、旧用户以及新用户的吞吐量则开始降低。

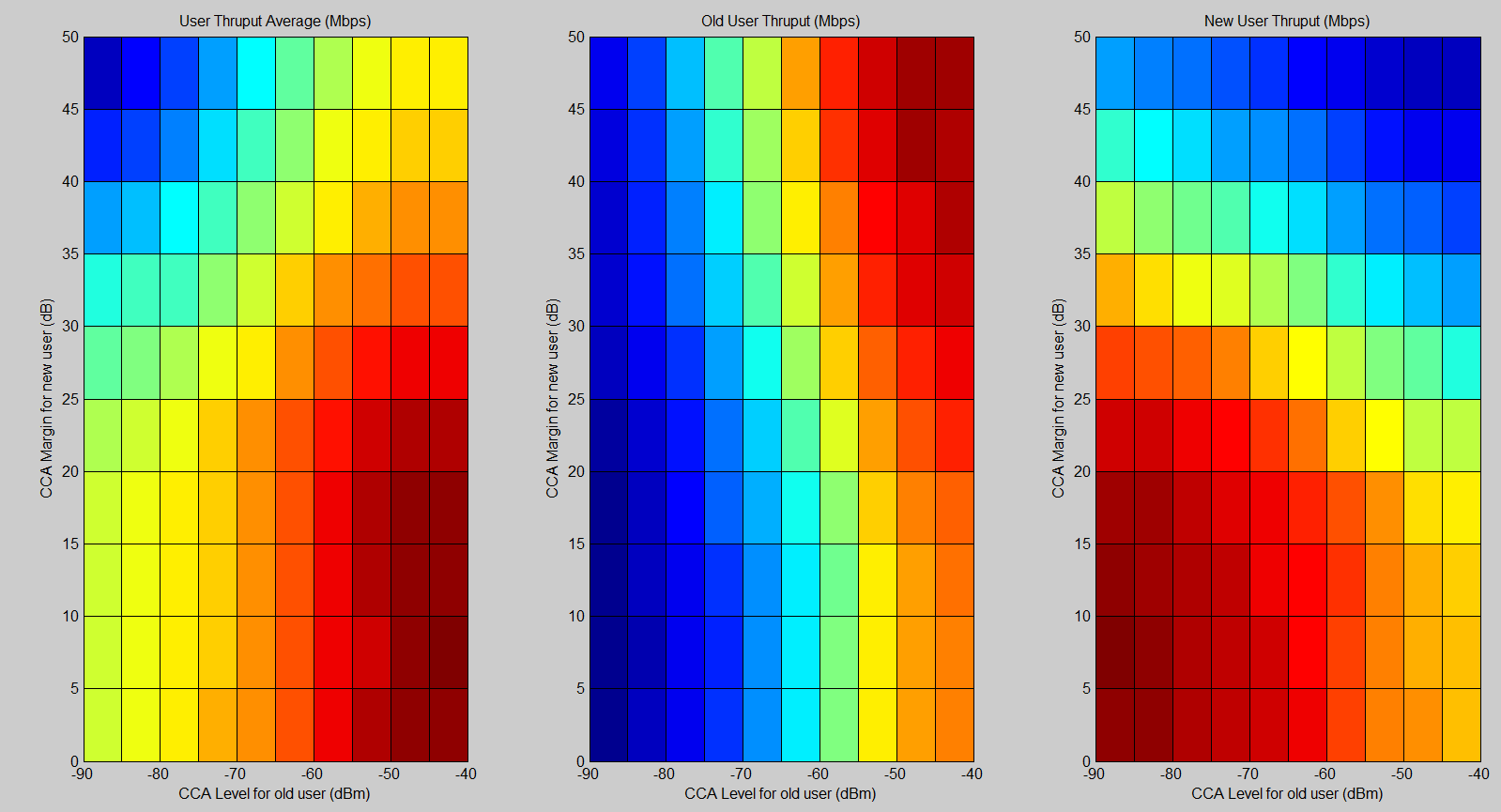
第三组：本组为纯静态CCA方案，新用户的CCA参数值与旧用户的参数值设置不相同，每个旧用户的CCA从-90dBm开始以5dBm为步长增加到-40dBm, 新用户CCA = 旧用户CCA +10 dBm，也就是说每个新用户的CCA Level则是从-80到-30dBm变化(横坐标)，而无论是新用户还是旧用户其TXP Level则是从3到40dBm(纵坐标)。具体的仿真结果如下图所示：



仿真结果分析：

对上图进行分析可以发现：无论是整个网络还是旧用户和新用户的吞吐量，其性能基本上都是一致的，可以表述为：随着CCA Level的增加，整个网络、旧用户以及新用户的吞吐量都是先增加后达到饱和；而随着TXP Level的增加，整个网络、旧用户以及新用户的吞吐量也是先增加到一定的值的时候当TXP Level再继续增加的时则开始降低。

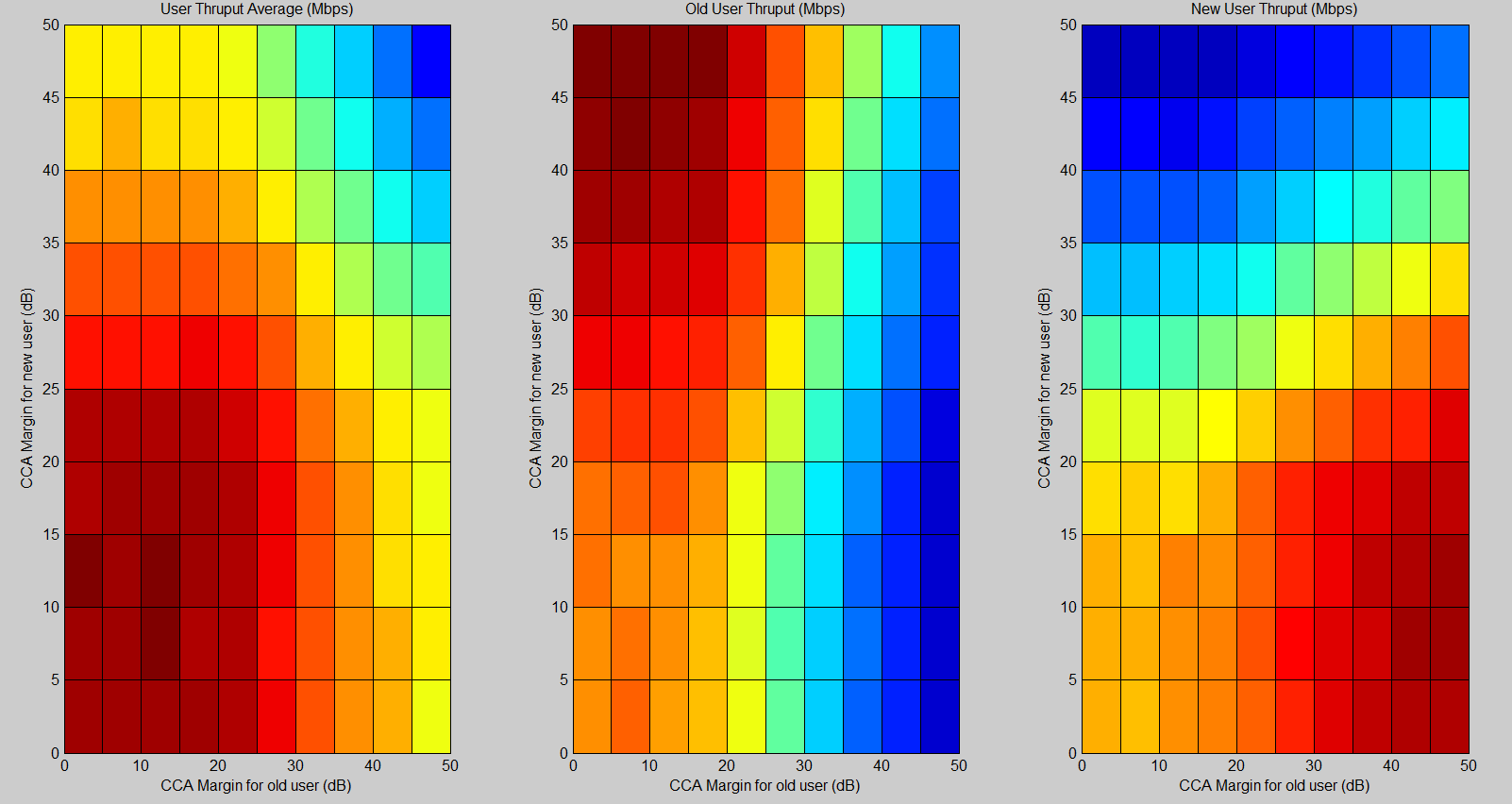
第四组：本组为半静半动的CCA方案，新用户的CCA参数值与旧用户的参数值设置不相同，每个旧用户的CCA Level是从-90dBm开始以5dBm为步长增加到-40dBm(横坐标), 每个新用户的CCA Margin则是从0dB开始以5dB为步长到50dB变化(纵坐标)，而无论是新用户还是旧用户其TXP Level均设置为固定的20dBm。具体的仿真结果如下图所示：



仿真结果分析：

对上图进行分析可以发现：随着旧用户CCA Level 值的增加，整个网络以及旧用户的的吞吐量先上升随后达到饱和，而新用户的吞吐量则是在降低的，而当随着新用户CCA Margin值的增加，整个网络和新用户的吞吐量则是在降低的而旧用户的吞吐量则是在上升的，并且当上升到一定的值的时候则不再上升。

第五组：本组为纯动态的CCA方案，新用户的CCA Margin值与旧用户的CCA Margin值设置不相同，每个旧用户的CCA Margin是从0dB开始以5dB为步长增加到50dB(横坐标), 每个新用户的CCA Margin则是从0dB开始以5dB为步长到50dB(纵坐标)，而无论是新用户还是旧用户其TXP Level均设置为固定的20dBm。具体的仿真结果如下图所示：



仿真结果分析：

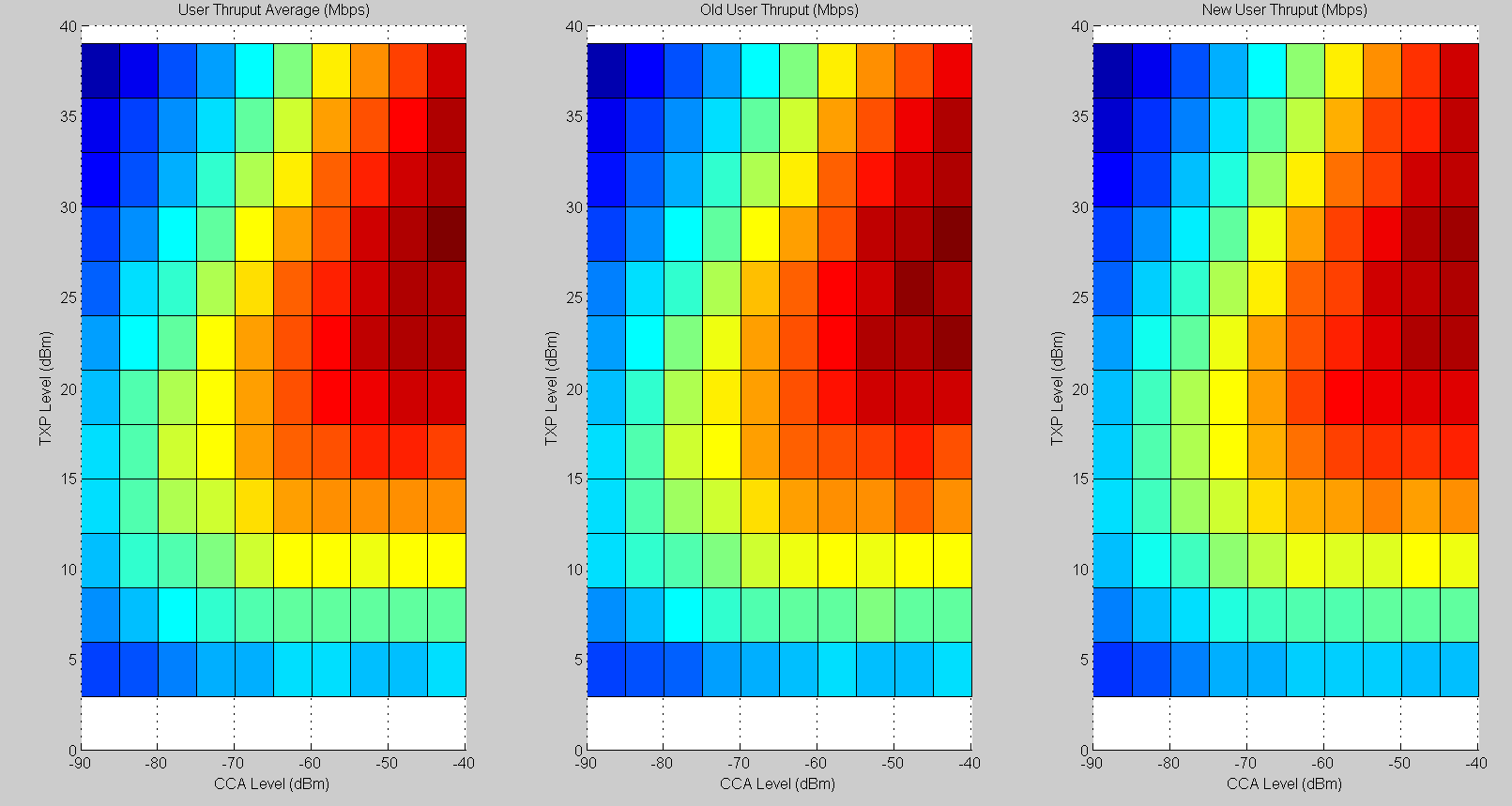
对上图进行分析可以发现：随着旧用户CCA Margin 值的增加，整个网络以及旧用户的的吞吐量先降低随后达到饱和，而新用户的吞吐量则是在增加的，而当随着新用户CCA Margin值的增加，整个网络和新用户的吞吐量则是在降低的而旧用户的吞吐量则是在上升的，并且当上升到一定的值的时候则不再上升。

### 仿真结果（频率复用因子为3）

仿真图说明：

在下文所述的所有仿真图中，自左至右分别表示的是整个网络的平均吞吐量（User Thruput Average），旧用户的吞吐量(Old User Thruput)，新用户的吞吐量(New User Thruput)。每张图中颜色的深浅代表吞吐量的高低，颜色越深说明吞吐量越大，颜色越浅说明吞吐量就越小，而图中具体的横坐标以及纵坐标的含义以及横纵坐标的取值范围将会在每张仿真图的分析中进行介绍。下面将进行五组仿真结果的具体分析：

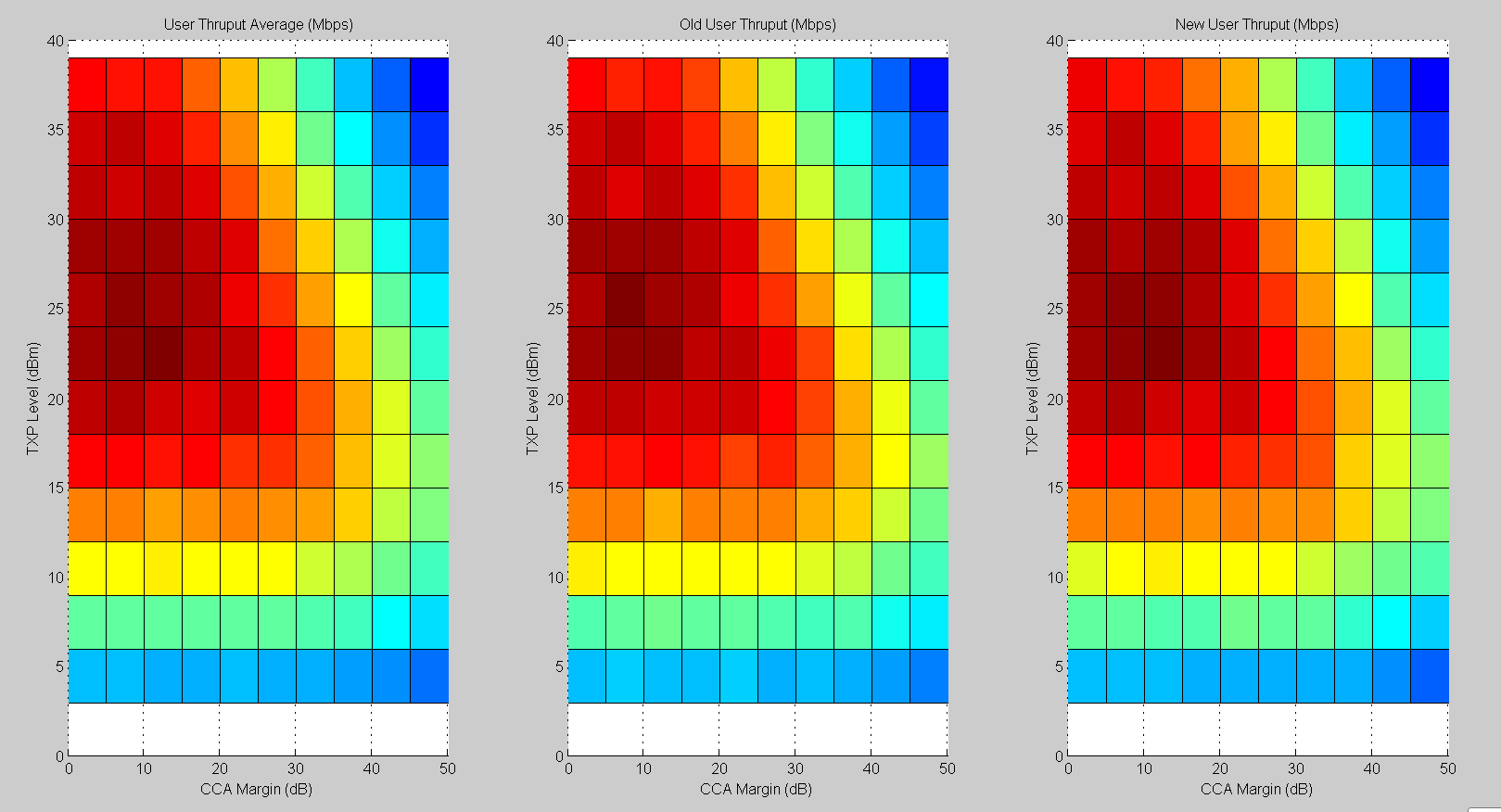
第一组：本组为纯静态CCA方案，新用户的CCA参数值与旧用户的参数值设置相同，新旧用户的CCA参数值均是从-90以5dBm为步长增加 到-40 dBm(横坐标),每个用户的TXP Level则是从3以3dBm为步长增加到40dBm(纵坐标)。具体的仿真结果如下图所示：



仿真结果分析：

对上图进行分析可以发现：无论是整个网络还是旧用户和新用户的吞吐量，其性能基本上都是一致的，可以表述为：随着CCA Level的增加，整个网络、旧用户以及新用户的吞吐量一开始都是增加随后达到饱和的；而随着TXP Level的增加，整个网络、旧用户以及新用户的吞吐量也是先增加随后达到饱和的。

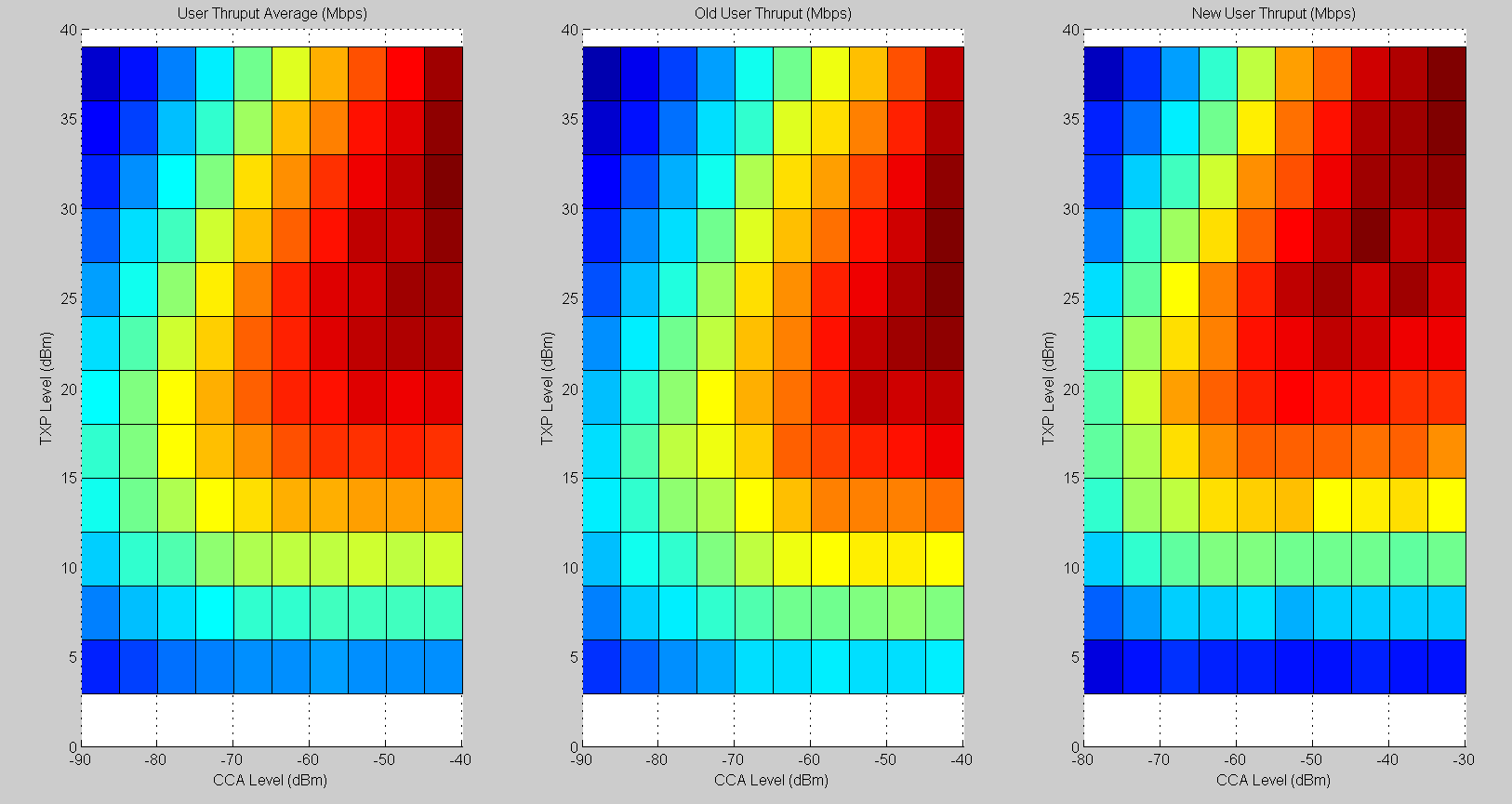
第二组：本组为纯动态CCA方案，新用户的CCA参数值与旧用户的CCA参数值设置相同，新旧用户的CCA Margin值均是从0 dB开始以5dB为步长增加到50 dB(横坐标),每个用户的TXP则是从3dBm开始以3dBm为步长增加到40dBm(纵坐标)。具体的仿真结果如下图所示：



仿真结果分析：

对上图进行分析可以发现：无论是整个网络还是旧用户和新用户的吞吐量，其性能基本上都是一致的，可以表述为：随着CCA Margin的增加，整个网络、旧用户以及新用户的吞吐量一开始都是慢慢降低，当降低到一定的程度的时候将不再降低；而随着TXP Level的增加，整个网络、旧用户以及新用户的吞吐量则是先增加到一定的值，而当TXP Level的值再继续增加的时候，整个网络、旧用户以及新用户的吞吐量则开始降低。

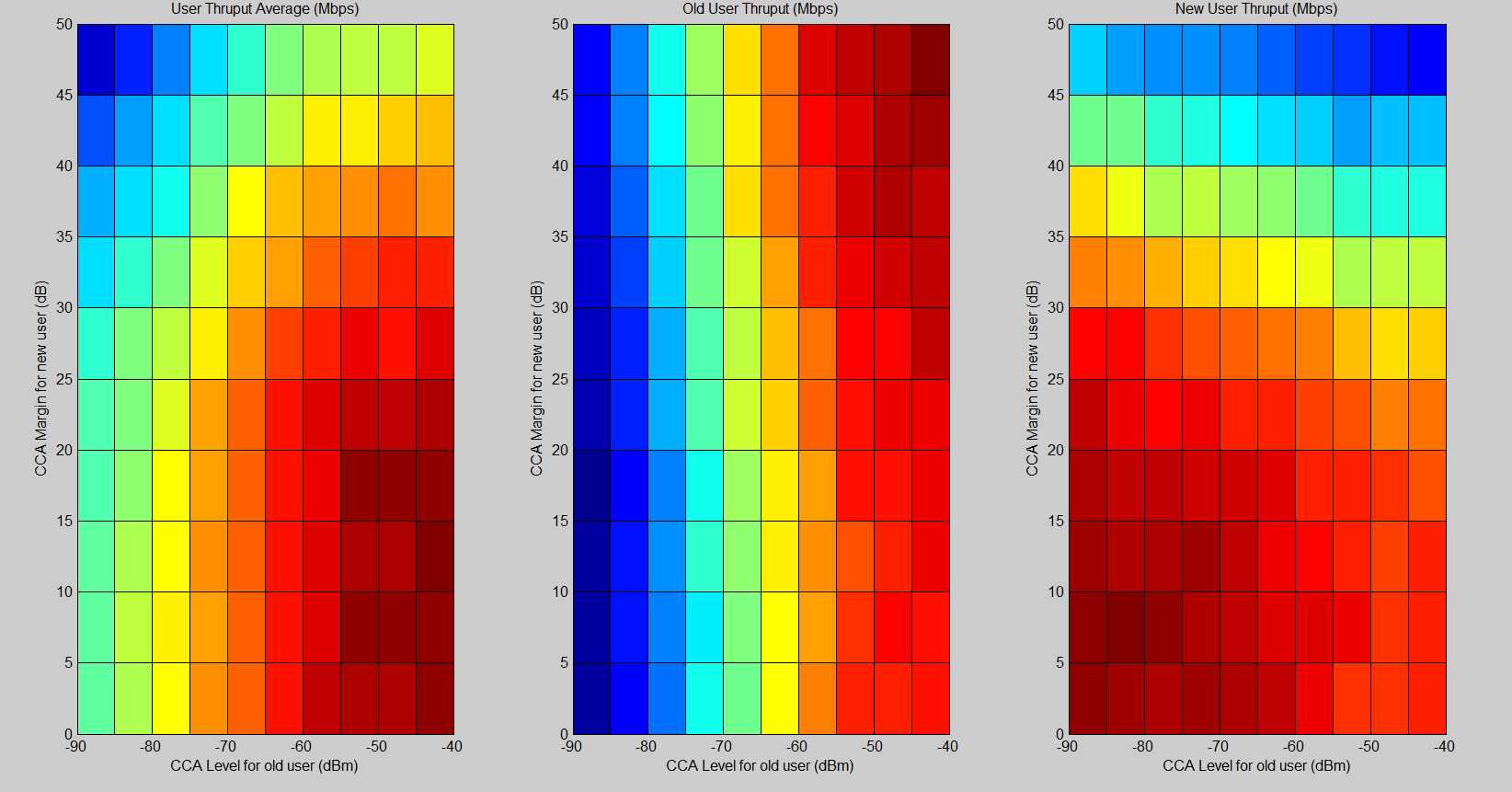
第三组：本组为纯静态CCA方案，新用户的CCA参数值与旧用户的参数值设置不相同，每个旧用户的CCA从-90dBm开始以5dBm为步长增加到-40dBm, 新用户CCA = 旧用户CCA +10 dBm，也就是说每个新用户的CCA Level则是从-80到-30dBm变化(横坐标)，而无论是新用户还是旧用户其TXP Level则是从3到40dBm(纵坐标)。具体的仿真结果如下图所示：



仿真结果分析：

对上图进行分析可以发现：无论是整个网络还是旧用户和新用户的吞吐量，其性能基本上都是一致的，可以表述为：随着CCA Level的增加，整个网络、旧用户以及新用户的吞吐量都是先增加后达到饱和；而随着TXP Level的增加，整个网络、旧用户以及新用户的吞吐量也是先增加到一定的值的时候当TXP Level再继续增加的时则开始降低。

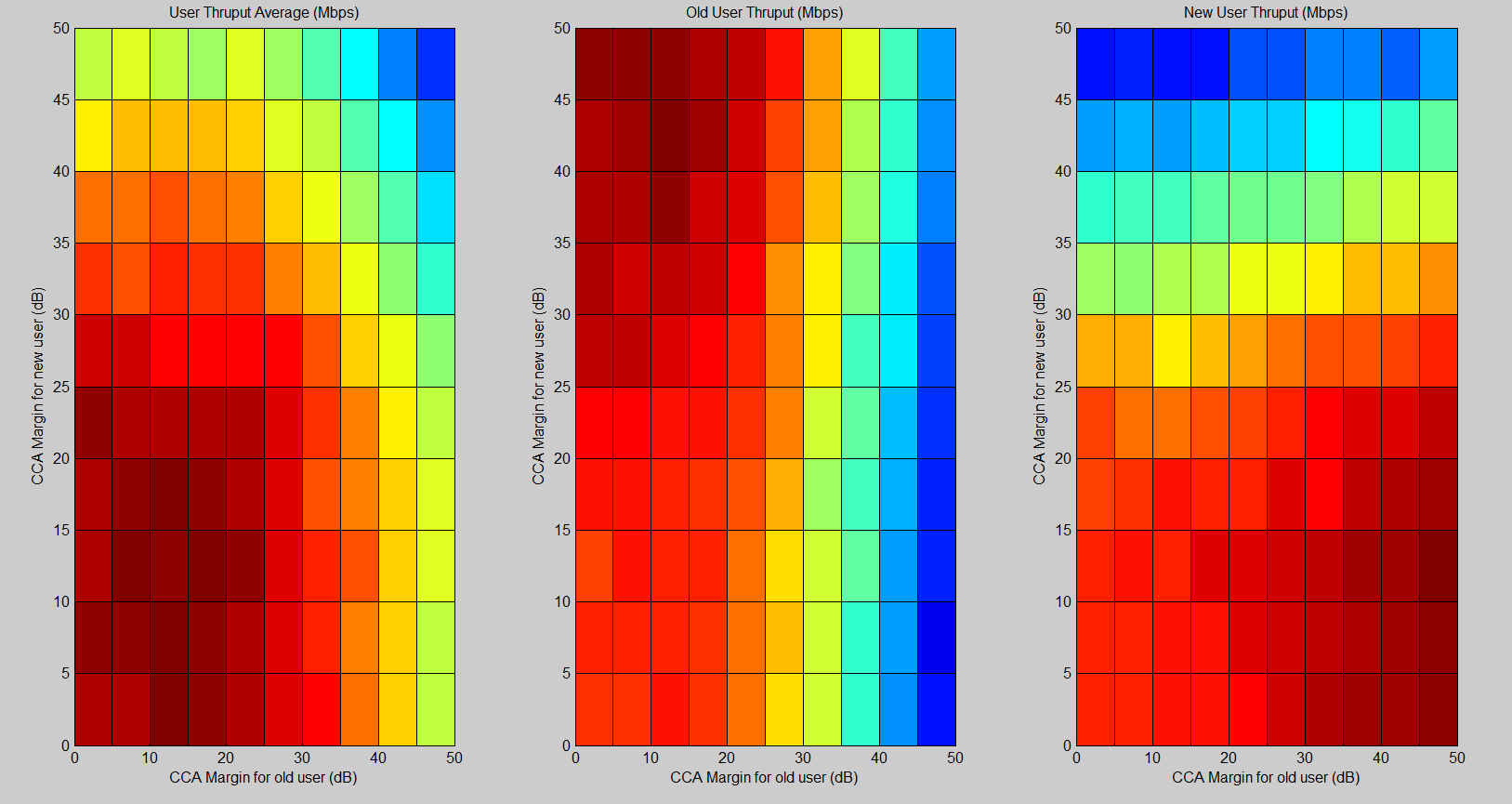
第四组：本组为半静半动的CCA方案，新用户的CCA参数值与旧用户的参数值设置不相同，每个旧用户的CCA Level是从-90dBm开始以5dBm为步长增加到-40dBm(横坐标), 每个新用户的CCA Margin则是从0dB开始以5dB为步长到50dB变化(纵坐标)，而无论是新用户还是旧用户其TXP Level均设置为固定的20dBm。具体的仿真结果如下图所示：



仿真结果分析：

对上图进行分析可以发现：随着旧用户CCA Level 值的增加，整个网络以及旧用户的的吞吐量先上升随后达到饱和，而新用户的吞吐量则是在降低的，而当随着新用户CCA Margin值的增加，整个网络和新用户的吞吐量则是在降低的而旧用户的吞吐量则是在上升的，并且当上升到一定的值的时候则不再上升。

第五组：本组为纯动态的CCA方案，新用户的CCA Margin值与旧用户的CCA Margin值设置不相同，每个旧用户的CCA Margin是从0dB开始以5dB为步长增加到50dB(横坐标), 每个新用户的CCA Margin则是从0dB开始以5dB为步长到50dB(纵坐标)，而无论是新用户还是旧用户其TXP Level均设置为固定的20dBm。具体的仿真结果如下图所示：



仿真结果分析：

对上图进行分析可以发现：随着旧用户CCA Margin 值的增加，整个网络以及旧用户的的吞吐量先降低随后达到饱和，而新用户的吞吐量则是在增加的，而当随着新用户CCA Margin值的增加，整个网络和新用户的吞吐量则是在降低的而旧用户的吞吐量则是在上升的，并且当上升到一定的值的时候则不再上升。

### 仿真结果对比（频率复用因子为1与频率复用因子为3的相同与不同）

相同点：

无论频率复用因子是1还是3，只要CCA的方案确定以后，整个网络以及新旧用户的吞吐量变化的趋势都是一样的。

不同点：

相同的CCA方案的情况下，当整个网络的CCA的值与TXP相同的时候，当复用因子为3的时候，整个系统、新用户以及旧用户的吞吐量都会比频率复用因子为1时整个系统、新用户以及旧用户的吞吐量要大。

### 结论

我们学习发射功率控制和空闲信道评估机制，为了有执照频谱网络LAA网络辅助进入无执照频谱。首先我们研究所有站点都配置相同的CCA值的空闲信道评估和所有站点都使用相同发射功率的静态发射功率控制。通过数值仿真的结果可以看出，一定的程度增加CCA参数的值和发射功率的值有助于提高整个系统的吞吐量和5%用户吞吐量。我们注意后者发现在最优CCA参数和最优TPC参数之间有一个有趣的线性关系。而且从观察整个网络吞吐量的点和5%用户吞吐量的点，发现这个最优CCA参数和发射功率是不相同的。

值得注意的是最优CCA和最优发射功率参数时依赖场景，比如房间的维度，穿透损耗，MCS表，频率复用因子等参数。自动选择CCA参数和发射功率参数的算法在实际LAA网络使用时非常重要的，我们将在不久的将来学习该算法。

其次，我们研究动态CCA和动态TPC策略，其中动态CCA是每一个站点都是用他们自己的CCA参数，动态发射功率控制是每一个站点都是使用自己的发射功率，发射功率是以服务基站的距离来制定的。CCA参数的值大约25dB与TPC参数大约为30dB通常会出现最优的网络吞吐量，但是在5%用户吞吐量中最优的情况看起来会更加复杂。不同的站点会有不同的CCA参数值，CCA参数值取决与服务基站的路径损耗。

# 参考文献

1. LTE: the UMTS long term evolution. New York: John Wiley & Sons, 2009.
2. Perahia, Eldad, and Robert Stacey. Next Generation Wireless LANS: 802.11 n and 802.11 ac. Cambridge university press, 2013.
3. Huawei Technologies, “The Unlicensed Spectrum Usage forFuture IMT Technologies.”
4. Ron Porat et. al., “Evaluation methodology”, IEEE 802.11-14-0980r5.
5. Simone etl. al., “Simulation Scenarios”, IEEE 802.11-14-0571r6.