中国研究生创新实践系列大赛

“华为杯”第二十届中国研究生数学建模竞赛

学　校	厦门大学
参赛队号	23103840031
队员姓名	1.陈一帆
	2.朱运俊
	3.陈宏毅

中国研究生创新实践系列大赛
“华为杯”第二十届中国研究生
数学建模竞赛

题 目　　WLAN 网络信道接入机制建模

摘　　要：

无线局域网（WLAN, Wireless Local Area Network）由于其共享信道和易受干扰的特点，可能导致网络中的节点在发送数据时发生碰撞、竞争以及丢包问题，从而显著降低系统吞吐等通信性能。因此，对无线局域网网络信道接入机制进行建模，并评估各个模型下的系统吞吐量，对于优化系统性能、提高通信服务质量具有重要意义。

本文从不同的多服务集（BSS，Basic Service Set）信道接入场景出发，基于马尔可夫模型及时序流程等进行分析，搭建不同场景下的信道接入机制模型，分析其数据传输过程，求解相应的模型参数，评估了系统吞吐量。并将模型推导出的系统吞吐量与仿真器模拟出的系统吞吐量进行对比，验证了模型的精确度。

对于问题一：其主要目标是评估存在同频干扰的 2BSS 场景的系统吞吐。我们首先以无线接入点（AP，Access Point）所处状态作为节点，建立了一个二维马尔可夫链并分析其状态转移概率。接着，我们根据马尔可夫链的常返性和稳态解的归一性等性质，求解了 AP 的发送概率和碰撞概率等模型参数，并基于此计算出了模型理论系统吞吐量，结果为 **67.1744Mbps**。最后，我们搭建了仿真器对所建模型的准确度进行了验证，仿真结果显示，我们的模型理论系统吞吐和仿真系统吞吐的误差为 0.5158%。

对于问题二：其主要目标是在问题 1 的基础上，评估在两个互听 AP 并发传输数据时均会成功的 2BSS 场景的系统吞吐。然而，尽管场景中的两个 AP 在发送数据时不会碰撞失败，但是在二进制随机退避阶段，需要考虑 AP 在侦听到信道忙时退避过程进入冻结状态的问题。针对上述情况，我们根据发送时隙的期望值、完整发送过程中时隙总数的期望值与节点发送概率之间的关系，计算节点的发送概率，并基于此计算出模型理论系统吞吐量，结果为 **72.6290Mbps**。最后，我们搭建了仿真器对所建模型的准确度进行了验证，仿真结果显示，我们的模型理论系统吞吐和仿真系统吞吐的误差为 6.541%。

对于问题三：其主要目标是评估两个不互听 AP 并发时会因交叠而导致失败的 2BSS 场景的系统吞吐。与问题 1、2 不同，问题 3 还考虑了复杂多变的无线传输环境引起的丢包问题。我们首先综合考虑了隐藏节点与环境因素建立了源节点二维马尔可夫模型，并据此推导出隐藏节点在脆弱期的发送概率（即与源节点并发交叠的概率）。之后，我们通过分析虚拟时隙分布来计算源节点本身的发送概率。并基于上述求解出的参数计算出了模型理论系统吞吐量。最后，我们搭建了仿真器对所建模型的准确度进行了验证。在题目（第一个结果）和附录 6（后六个结果）提供的 7 种不同场景下，我们的模型理论系统吞吐分别为 **52.7934Mbps, 46.4550Mbps, 40.8410Mbps, 46.8293Mbps, 37.9241Mbps, 33.5274Mbps, 38.3632Mbps**，和仿真系统吞吐的误差最大为 3.685%，最小为 1.951%。

对于问题四：其主要目标是评估三个部分互听的 AP 组成的 3BSS 场景的系统吞吐。在题目条件中，AP1 和 AP3 之间不互听，AP2 与其余两者都互听，且 AP1 与 AP3 并发传

输时均会成功。由上述条件易知，AP1 和 AP3 的发送概率和碰撞概率都应是相等的，所以只需要分析 AP1 和 AP2 的发送概率与碰撞概率。类似问题 1，我们根据二维马尔可夫链，推导出了 AP1 和 AP2 各自的发送与碰撞概率关系，得到两组方程。根据互听条件，我们通过 AP1 的发送概率计算 AP2 的碰撞概率，由 AP2 的发送概率计算 AP1 的碰撞概率，得到两组方程。之后我们通过求解上述 4 组方程，得到不同 AP 节点的发送概率，并基于此计算出了模型理论系统吞吐量。最后，我们搭建了仿真器对所建模型的准确度进行了验证。在题目和附录 6 提供的 7 种不同场景下，我们的模型理论系统吞吐分别为 **108.4228Mbps, 98.7974Mbps, 82.8179Mbps, 98.8140Mbps, 82.7612Mbps, 71.2892Mbps, 82.7761Mbps**，和仿真系统吞吐的误差最大为 5.352%，最小为 0.4720%。

关键词：WLAN 信道接入机制 同频干扰 并发传输 马尔可夫模型

目录

一、问题重述

1.1 问题背景

随着各种移动终端设备的普及以及移动互联网的飞速发展，无线局域网因其具有吞吐量高、成本低、布网简单等优势得到了人们的青睐。IEEE 802.11 系列协议广泛应用于无线局域网、无线自组网络的理论研究、仿真研究以及各类试验床。其中分布式协调功能（DCF，Distributed Coordination Function）作为介质访问控制子层（MAC ，Medium Access Control）中最基本的无线信道接入方式，采用载波侦听多址接入/退避的随机接入协议，并在发生碰撞后采用二进制指数退避算法避免再次冲突，对 DCF 进行数学建模和性能分析已经成为计算机通信领域的研究热点。

在饱和状态条件下，Bianchi[1-2]提出了基于二维马尔科夫链对 IEEE 802.11 DCF 协议建模，在理想信道状态下，通过对该马尔可夫模型的稳态求解，得到任意时隙内节点发送概率以及与其他节点的冲突概率，从而导出系统中主要的性能指标——饱和吞吐量；在 Bianchi 工作的基础上，Chatzimisios[3]和 Kuo[4]分别增加了最大重传次数的限制和考虑了冻结状态的因素；考虑到实际网络中语音业务或数据业务常常导致终端 MAC 层队列为空，即网络处于非饱和状态，Duffy 等人[5–7]对 Bianchi 模型的状态空间进行扩展，增加一个或多个空闲状态以表示终端的非饱和状态，并通过求解二维马尔可夫模型实现对 DCF 的性能分析；ZHAI[8]用状态转移函数推导出一个饱和状态下的系统层服务时间的近似分布函数模型，同时结合 M/G/1/K 排队模型和 M/M/1/K 队列模型分析了非饱和条件下的吞吐量和时延。

然而电磁波信号随距离能量衰减严重，因此无线接入点（AP，Access Point）仅能通过侦听信号强度判断周围是否存在同频干扰，在 AP 密集部署时，同频 AP 之间的距离远，接收信号强度低于门限，此时 AP 认为信道空闲，因此总是在退避和发送数据；无线信号传输环境是复杂多变的，当有遮挡物或者人走动时，无线信道可能会快速发生较大的变化，从而导致不同程度的丢包；在多服务集系统中，AP 间可能存在部分互听，部分不互听的情况；多个 AP 同时发送数据时，由于信干比不同，碰撞导致的传输失败率几率也不同。因此，如何依据实际情况构建更加准确的 DCF 模型并分析性能是亟需解决的问题。

1.2 问题提出

基于上述背景，题目提供了 Bianchi 模型的建模及求解方法，本题需要解决以下问题：

问题一：考虑两个 BSS 互听的下行通信系统，大部分时候只有一个 AP 能够接入信道，数据传输一定成功。然而，当两个 AP 同时回退到 0 而同时发送数据时，存在同频干扰，导致两个 AP 的数据都发送失败。要求对该系统进行建模，并用数值分析方法求解，评估系统的吞吐，同时采用仿真器验证模型精确度。

问题二：在问题 1 的系统的基础上降低两个 AP 传输时的物理层速率，使并发时两个终端接收到数据的信干比（SIR， Signal-to-Interference Ratio）升高，从而使两个 AP 的数据传输都能成功。要求对该系统进行建模，并用数值分析方法求解，评估系统的吞吐，同时采用仿真器验证模型精确度。

问题三：在两个不互听的 BSS 中，考虑实际场景中有遮挡物或者有人走动时，无线信道可能会快速发生较大的变化，即便不存在邻服务集的干扰，AP 发送数据也会有 10%概

率的丢包。且当两个 AP 发包在时间上有交叠时，会因 SIR 较小导致两者发包均失败。要求对该系统进行建模，并用数值分析方法求解，评估系统的吞吐，同时采用仿真器进行验证。

问题四：考虑包含 3 个 BSS 的下行通信系统，其中 AP1 与 AP2 之间，AP2 与 AP3 之间均互听，AP1 与 AP3 之间不互听，且当 AP1 和 AP3 的发包时间交叠时，会因 SIR 较大导致两者发送均成功；当 AP2 与 AP1 或 AP3 之间发包时间交叠时，则会由于同频干扰会导致发送失败。要求对该 3BSS 系统进行建模，用数值分析方法求解，评估系统的吞吐，同时采用仿真器进行验证。

二、问题分析

对于问题一，我们需要考虑题目提供的 2BSS 系统中两个 AP 同时发送所导致的数据包发送失败的问题，可以根据 AP 所处状态节点建立二维马尔可夫链，以求解 AP 的发送概率和碰撞概率，之后再进一步计算系统吞吐量。在建立二维马尔可夫链部分，既要考虑状态转移概率与发送概率之间的关系，还要考虑发送时碰撞与不碰撞这两种互斥事件的概率关系。在吞吐量计算部分，需要根据在上一部分中所求解的节点在每一时隙发送的概率来计算每一时隙至少有一个节点发送数据包的概率；然后在节点发送数据包的基础上，根据节点发包成功的条件，计算发送成功的概率。最后根据每一时隙有发送的概率、发送成功的概率、三种不同时隙的长度和有效载荷的长度计算出发送有效载荷的时间占比，再根据物理层传输速率计算出系统吞吐量。

对于问题二，我们需要在问题一的基础上考虑两个 AP 并发时由于终端 SIR 较高使得数据传输均能成功的条件，需要根据新的条件计算节点的发送概率和计算系统吞吐量。相比于问题一，本问题不用考虑节点发送数据时的碰撞问题，节点发送数据时的成功概率为 100%，但是不能因此就将其考虑为两个互不相关单 AP 系统，简单地将吞吐量计算问题变成两个单 AP 系统吞吐量的求和。这是因为在本问题中，两个 AP 虽然在发送数据时不会因为碰撞导致失败，但是节点在二进制随机退避阶段，需要考虑在侦听到信道忙时（即另一个节点发送数据时），退避过程进入冻结状态的问题。所以在计算节点发送概率阶段，我们考虑一个完整的发送过程中空闲时隙数与发送时隙数的占比，对于其中一个节点来说，其在退避阶段进入冻结状态意味着另一个节点正在发送数据，所以只有两个节点同时退避的时隙才能算作空闲时隙。据此，我们根据发送时隙数的期望值与完整发送过程中时隙总数的期望值的占比等于发送概率这一条件，可列出方程求解 AP 的发送概率。在计算吞吐量部分，我们根据 AP 的发送概率计算发送有效载荷时长在总时长中的占比，从而计算系统的吞吐量。但需要注意在部分发送时隙中，两个节点会同时发送数据，所以对于这一部分发送时隙，我们需要额外计算一份有效载荷。

对于问题三，我们需要考虑 2BSS 系统中隐藏节点的问题，并根据题目提供的条件对虚拟时隙分布进行分析，并且结合并发传输交叠和通信环境差导致丢包的条件建立二维马尔可夫模型分析节点的状态变化，从而得出节点的发送概率，进而计算系统的吞吐量。在考虑源节点发送数据包期间（脆弱期），隐藏节点同时发送数据导致并发传输交叠的概率时，我们参考了 Huang 和 Ivan Marsic（2010）[9]的模型建立了源节点的二维马尔可夫链。但该模型没有考虑环境原因引起的丢包率，所以我们在其基础上考虑了环境因素引起的丢包重传，建立了更为准确的二维马尔可夫模型，以推导出隐藏节点在脆弱期的发送概率。由于问题中的两个节点不互听，所以在考虑源节点本身在每个虚拟时隙的发送概率时，我们选择性忽略了隐藏节点，通过分析不同虚拟时隙的概率分布来计算节点本身的

发送概率。最后通过节点本身的发送概率以及隐藏节点在脆弱期的发送概率计算系统的吞吐量。

对于问题四，我们需要考虑更为复杂的 3BSS 系统，并且结合题目给的 AP 之间不同的互听关系建立两个二维马尔可夫链，以分析两种 AP 的发送概率，进而计算系统的吞吐量。由于题目所给的不是最简单的 3AP 互听的 3BSS 系统，需要考虑 AP1 和 AP3 之间不互听，AP2 与其余两者都互听的关系，所以不能简单地将 3 个 AP 的发送概率按相等关系来考虑。由于 AP1 和 AP3 的条件一致，且互不影响发送数据，所以这两者的发送概率和碰撞概率都应是相等的，且碰撞概率可由 AP2 的发送概率得出，而 AP2 的碰撞概率则可由 AP1 和 AP3 的发送概率得出。对于 AP1 和 AP3 的发送概率和碰撞概率之间的关系，可以建立二维马尔可夫链得到相应的关系方程。同理可得到 AP2 的发送概率和碰撞概率关系方程。根据上述的概率关系可以得到关于 AP1（AP3 同 AP1）的发送概率和碰撞概率、AP2 的发送概率和碰撞概率的 4 个方程，根据题目所给条件，联立即可解出这 4 个概率。在得到不同 AP 节点的在每个时隙的发送概率之后，再注意到部分时隙中 AP1 和 AP3 并发传输时需要额外计算一份有效载荷的情况，即可计算系统吞吐量。

三、模型假设

假设 1： 假设评估的系统吞吐量均为饱和吞吐量
假设 2： 假设节点传输过程中 ACK 不会丢失
假设 3： 采用 AP 的工作模式均采用标准模式

四、符号系统

符号	含义
S	饱和状态下的吞吐量
n	网络中节点数量
τ	节点发送数据概率
p	节点碰撞概率
σ	退避阶段的时隙长度
r	最大重传次数
m	最大退避阶数
$b(t)$	t 时刻一个节点退避随机过程的退避计数
$s(t)$	t 时刻一个节点退避随机过程的退避阶数
W_i	i 阶竞争窗口大小

l	AP 发送包的载荷长度
pl	PHY 头时长
v	物理层速率
t	一个时隙内传输的有效载荷发送时长
ts	一个时隙长度
p_f	失败重传概率
H	数据帧头
$E[P]$	数据帧的有效载荷传输时长
$E^*[P]$	发生冲突时较长数据帧的有效载荷传输时长
T_e	空闲时隙长
T_s	发送时隙长
T_c	碰撞时隙长
N_e	空闲时隙个数
CW_{min}	最小竞争窗口大小
CW_{max}	最大竞争窗口大小

五、问题一的建模与求解

5.1 解题思路与建模

本题的整体解题思路和建模流程如图 5-1 所示。

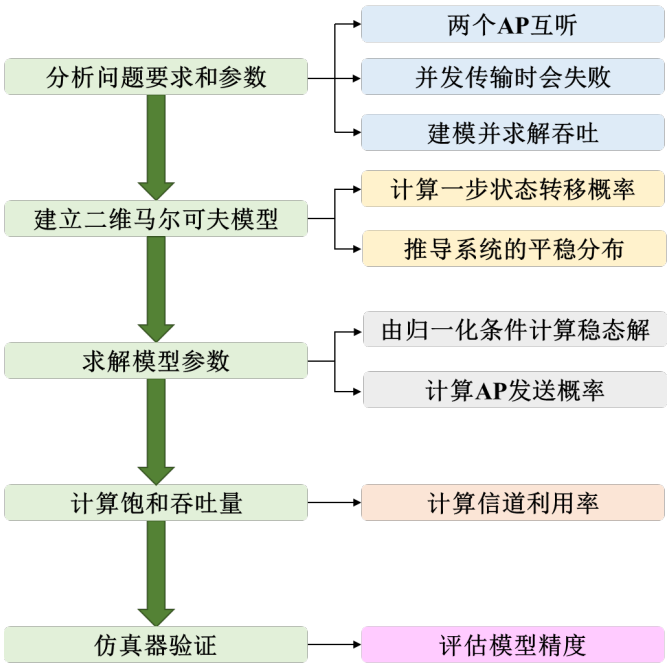


图 5-1 问题 1 的整体思路流程

如图 5-2 所示，本题要求我们在两个 BSS 互相监听，并且碰撞时会产生同频干扰而导致两个 AP 的数据传输失败的场景下，求解吞吐量。据此，我们首先以 AP 所处状态为节点建立了二维马尔可夫模型，计算了马尔可夫链的一步状态转移概率并以此推导了系统的平稳分布，之后由归一化条件，我们计算出了马尔可夫模型的稳态解，获取了马尔可夫模型的相关参数，并计算了系统吞吐量。最后，我们基于上述场景编写了仿真器来评估所提模型的准确度。

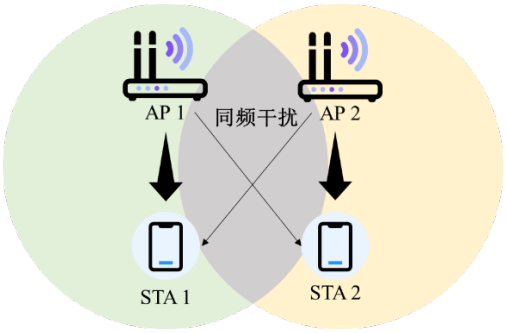


图 5-2 问题一场景示意图

5.2 数据传输碰撞概率分析

如图 5-3 中的所示，问题 1 存在的两个 AP 同时回退到 0 而同时发送数据时产生同频干扰而导致数据传输失败的场景，根据 Bianchi 模型，竞争窗口的大小为：

$$W_i = \begin{cases} 2^i W_0, 0 \le i \le m \\ 2^m W_0, m \le i \le r \end{cases} \tag{5-1}$$

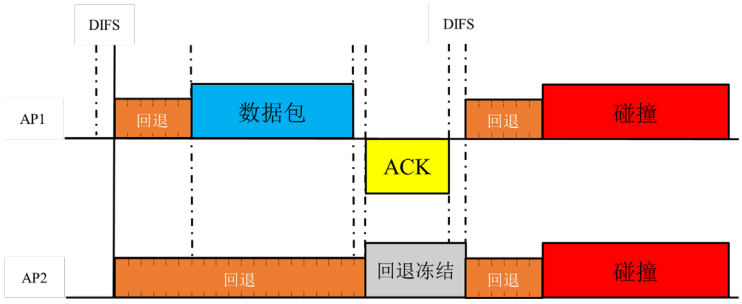


图 5-3 两个 AP 碰撞过程示例

可将 AP 状态$\{b(t),s(t)\}$的转移过程表示为如图 5-4 所示的二维马尔可夫过程：

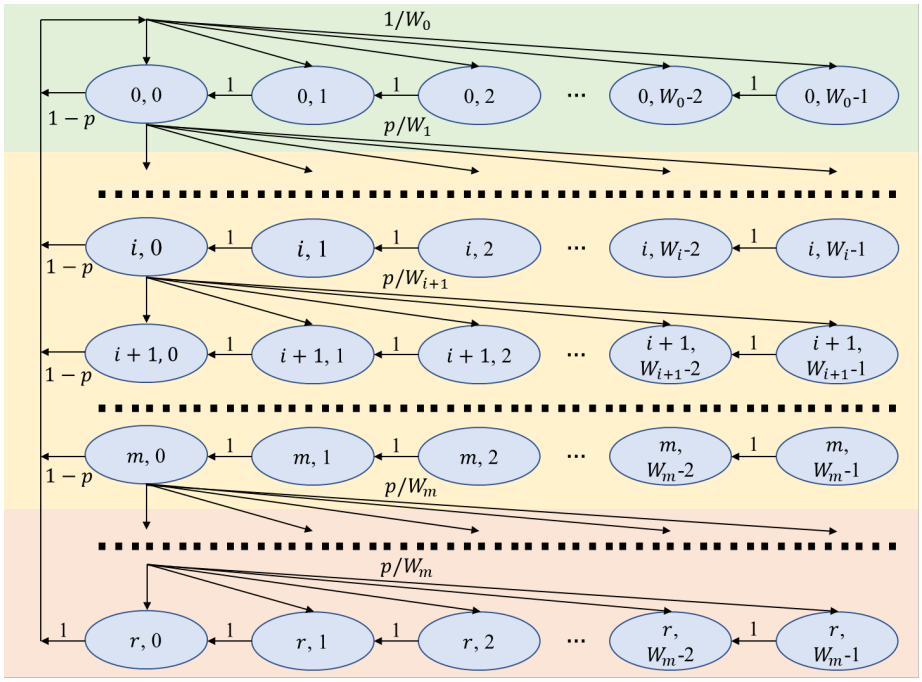


图 5-4 问题 1 马尔可夫过程

由图 5-4 可以得到以下的一步转移概率:

$$P(i,k\,|\,i,k+1)=1,k\in[0,W_i-2],i\in[0,r] \tag{5-2}$$

$$P(0,k\,|\,i,0)=\frac{1-p}{W_0},k\in[0,W_0-1],i\in[0,r] \tag{5-3}$$

$$P(i,k\,|\,i-1,0)=\frac{p}{W_i},k\in[0,W_i-1],i\in[1,r] \tag{5-4}$$

$$P(0,k\,|\,r,0)=\frac{1}{W_0},k\in[0,W_0-1] \tag{5-5}$$

其中，式(5-2)表示了一次回退的情况，由于其不存在碰撞，所以概率为 1。式(5-3)表示当节点回退到 0 时，若发送成功（概率$1-p$），则需要重新在W_0窗口内选择一个随机数（即某个随机数被选中的概率为$\frac{1-p}{W_0}$），并转移到该状态。式(5-4)表示当在第$i-1$个回退

阶段，节点回退到 0 时，若发送失败（即概率 p），则需要首先调节窗口大小（以 2 为指数增长），即 W_{i-1} 变为 W_i。然后节点需要在更新后的窗口大小 W_i 中选择一个随机数（概率为 $\frac{p}{W_i}$），并转移至该状态。式(5-5)表示当节点到达最大的重传次数以后，无论成功还是失败，窗口都会重置，节点需要在重置后的窗口大小 W_0 中选择一个随机数（概率为 $\frac{1}{W_0}$），并转移至该状态。

该二维马尔可夫模型稳态下的概率分布为:

$$b_{i,k}=\lim_{t\to\infty}P\{s(t)=i,b(t)=k\},i\in(0,m),k\in(0,W_i-1) \tag{5-6}$$

对 W_i 种发送失败的马尔可夫链的状态转移步长求和可得:

$$b_{i,k}=\frac{W_i-k}{W_i}*b_{i,0}\qquad 0\le i\le r,0\le k\le W_i-1 \tag{5-7}$$

即:

$$b_{i,0}=p^i*b_{0,0},0<i\le r \tag{5-8}$$

对于发送的状态（即 k=0），则必然是从任意退避阶段退避数变为 0 时发送，或达到重传次数限制后转移而来的。因此，我们有:

$$b_{i,k}=\begin{cases}b_{i-1,0}*p*\dfrac{W_i-k}{W_i},0<i<r\\(1-p)*\dfrac{W_i-k}{W_i}*\sum_{j=0}^{r}b_{j,0},i=0\end{cases} \tag{5-9}$$

联立(5-8)和(5-9)式，可得：

$$b_{i,k}=\frac{W_i-k}{W_i}*b_{i,0}\qquad 0\le i\le r,0\le k\le W_i-1 \tag{5-10}$$

由于稳态分布的归一化，各个状态的概率总和为 1，所以有：

$$1=\sum_{k=0}^{W_i-1}\sum_{i=0}^{r}b_{i,k}=\sum_{i=0}^{r}b_{i,0}\sum_{k=0}^{W_i-1}\frac{W_i-k}{W_i}=\sum_{i=0}^{r}b_{i,0}\frac{W_i+1}{2} \tag{5-11}$$

联立式子(5-1)和(5-11)，可求得：

$$b_{0,0}=\begin{cases}\dfrac{2(1-p)(1-2p)}{(1-2p)(1-p^{r+1})+W_0(1-p)(1-(2p)^{r+1})},r\le m\\\dfrac{2(1-p)(1-2p)}{W_0(1-(2p)^{m+1})(1-p)+(1-2p)(1-p^{r+1})+W_0 2^m p^{m+1}(1-p^{r-m})(1-2p)},m<r\end{cases} \tag{5-12}$$

由公式(5-8)和(5-9)可易知，退避过程中每个退避状态均可由 $b_{0,0}$ 表示，故由公式(5-8)可求得模型中每个退避状态的值。

当节点退避到 0 时开始发送数据，因此节点在一个时隙发送数据帧的概率为

$$\tau = \sum_{0}^{r} b_{i,0} = b_{0,0} * \frac{1-p^{r+1}}{1-p} \tag{5-13}$$

当传输数据冲突，导致碰撞时，有:

$$p = 1-(1-\tau)^{N-1} \tag{5-14}$$

针对问题 1 中两个 AP 同时发送数据导致碰撞的情况，根据通用参数列表，易得 $m < r$，根据(5-12)可得:

$$b_{0,0} = \frac{2(1-p)(1-2p)}{W_0(1-(2p)^{m+1})(1-p)+(1-2p)(1-p^{r+1})+W_0 2^m p^{m+1}(1-p^{r-m})(1-2p)} \tag{5-15}$$

联立式子(5-13)，(5-15)可得：

$$\tau = \frac{2(1-2p)(1-p^{r+1})}{W_0(1-(2p)^{m+1})(1-p)+(1-2p)(1-p^{r+1})+W_0 2^m p^{m+1}(1-p^{r-m})(1-2p)} \tag{5-16}$$

联立(5-14)，(5-16)，根据题目所给条件，令 r=32， r=6， W_0=16，可解得:

$$\tau = p = 0.1046$$

5.3 吞吐量性能分析

根据 Bianchi 提出的模型，饱和状态下吞吐量的表达式为：

$$S = \frac{E[t]}{E[ts]} * v \tag{5-17}$$

其中分子部分表示一个时隙内有效载荷传输的平均时间，分母部分表示一个时隙的平均时间。

为了计算吞吐量 S，首先需要分析在一个时隙的时间内可能发生的情况。令 P_{tr} 为在考虑的时隙时间内至少有一个 AP 发送数据的概率，共有 n 个 AP 在竞争信道，每个 AP 以概率 τ 来进行发送速率，可得至少有一个 AP 发送数据的概率 P_{tr} 为：

$$P_{tr} = 1-(1-\tau)^{n} \tag{5-18}$$

其中，$(1-\tau)$ 代表其中某一个节点没有发送数据的概率，则 $(1-\tau)^n$ 即为所有节点都不发送数据的概率，所以其互斥事件的概率 $1-(1-\tau)^n$ 即为至少有一个节点发送的概率。

而由于在 Bianchi 提出的模型中，仅当有且只有一个节点发送时，才不会发生冲突。所以在有节点发送(即 $P_{tr} \neq 0$)的前提条件下下, 成功传输的概率为:

$$P_s = \frac{n\tau(1-\tau)^{n-1}}{P_{tr}} = \frac{n\tau(1-\tau)^{n-1}}{1-(1-\tau)^{n}} \tag{5-19}$$

其中分子中的 τ 代表有一个节点正在发送，$(1-\tau)^{n-1}$ 代表其余 $n-1$ 个节点都没有发送，而成功发送仅需某一个节点发送即可，并不需要确定的某个节点，所以分子再乘以节点

数 n 。

所以，在一个时隙时间内成功传输有效载荷的时间为：

$$I = E[P]P_{tr}P_s \tag{5-20}$$

而一个时隙的时间中一般包括三种情况：

情况 1：若节点在回退过程中，经过一个时隙，回退计时器未倒数到 0，其概率为 $1-P_{tr}$，消耗的时间就是一个时隙，即 σ 时间。而若节点经过这个时隙，回退计时器正好倒数到 0，那么节点就会发送数据（即概率 P_{tr} ）。但是发送后，这里还存在两种情况：发送成功或发送失败。

情况 2:若节点成功发送该数据，那么所花费时间为 T_s，其对应概率为 $P_{tr}P_s$，即成功发送的概率。

情况 3 :若节点发送数据，但是由于碰撞冲突，这个数据帧是无法成功被接收的，这个损耗的时间就是 T_c。其对应概率为 $P_{tr}(1-P_s)$，即发送失败的概率。

所以，饱和状态下吞吐量的表达式为：

$$S = \frac{P_s P_{tr} E[P] v}{(1-P_{tr})\sigma + P_{tr}P_s T_s + P_{tr}(1-P_s)T_c} \tag{5-21}$$

其中，成功传输和发生碰撞的传输时长 T_s 和 T_c 分别表示为:

$$T_s = H + E[P] + SIFS + ACK + DIFS \tag{5-22}$$

$$T_c = H + E^*[P] + SIFS + ACKTimeout \tag{5-23}$$

其中，由于题目中所给 AP 发送包的载荷长度为定值，所以在本题中 $E[P] = E^*[P]$，而数据帧头传输时间 H 包括了 MAC 层头传输时间 T_M 和物理层头传输时间 T_P，即：

$$H = T_M + T_P \tag{5-24}$$

据题目所给条件，联立(5-21)，(5-22)，(5-23)，(5-24)，令 $v = 455.8$ Mbps, $T_M = \frac{30*8}{455.8} = 0.5265\mu s$, $T_P = 13.6\mu s$, $E[P] = E^*[P] = \frac{1500*8}{455.8} = 26.3273$ μs, $ACK = 32\mu s$， $SIFS = 16\mu s$， $DIFS = 43\mu s$， $\sigma = 9\mu s$， $ACKTimeout = 65\mu s$，可解得:

$$S = 67.1744\text{Mbps}$$

5.3 仿真器性能验证

为了验证所建模型的精确度，我们使用 Python 语言，按照如图 5-5 所示的流程编写了仿真器。首先，我们参照题目以及附录 4 设置了物理层速率、载荷长度等仿真参数，接着进入数据传输过程，AP 首先检测信道的忙闲状态，若信道空闲且持续一个 DCF 帧时间，则终端随机地在 $[0, W_i - 1]$ 之间均匀选取一个回退时间并赋值给回退计数器，并同时检测信道是否存在同频干扰，若不存在同频干扰且信道空闲持续一个退避时隙时，计数器减一；

否则进入传输冻结阶段，计数器不变，在信道空闲时间等于 DIFS 时，重新激活延时计数器并继续监听信道状态，直到计数器减到 0 时才进行数据传输。否则退避阶数加一，以回到随机回退状态。

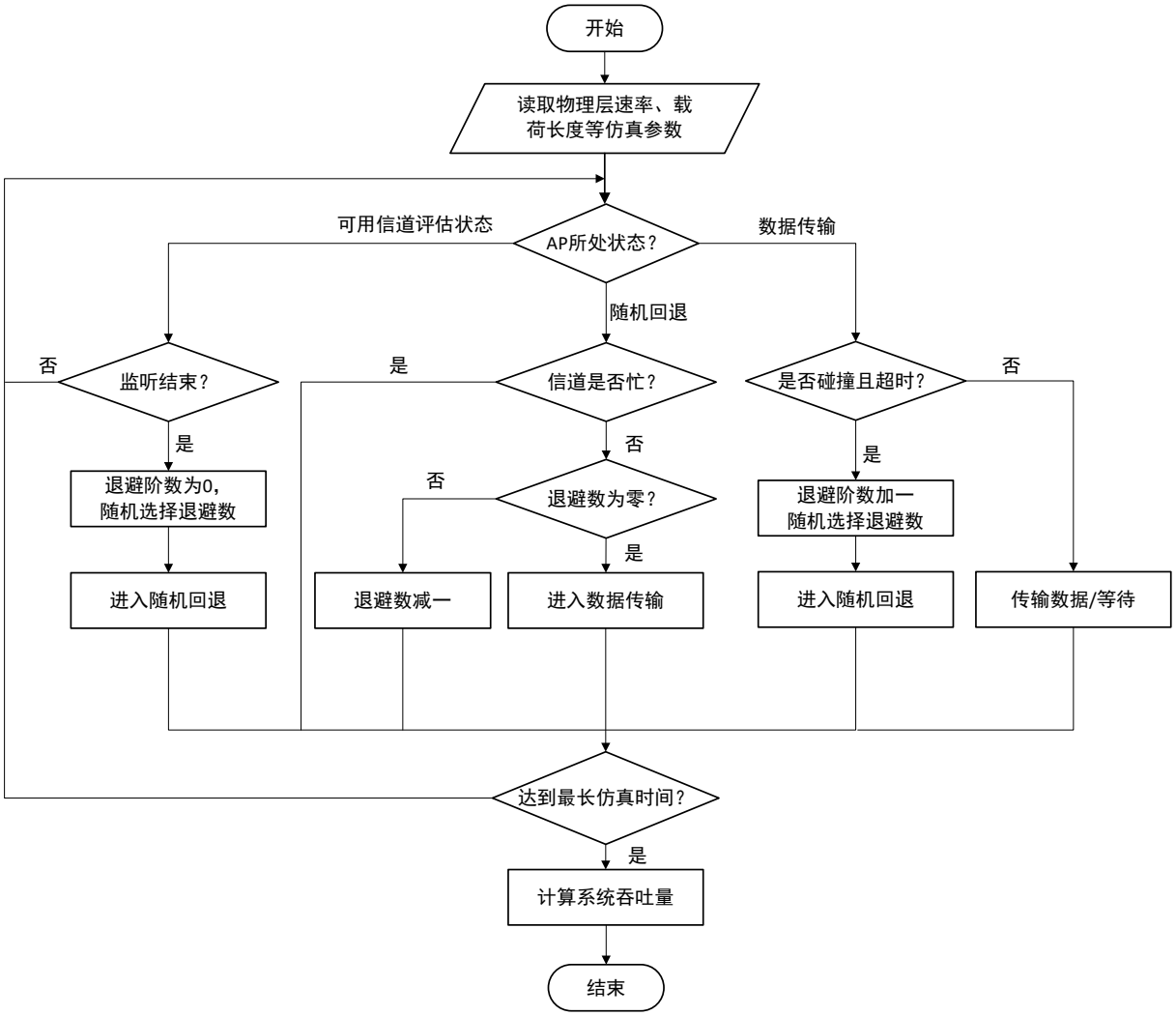


图 5-5 问题 1 仿真流程图

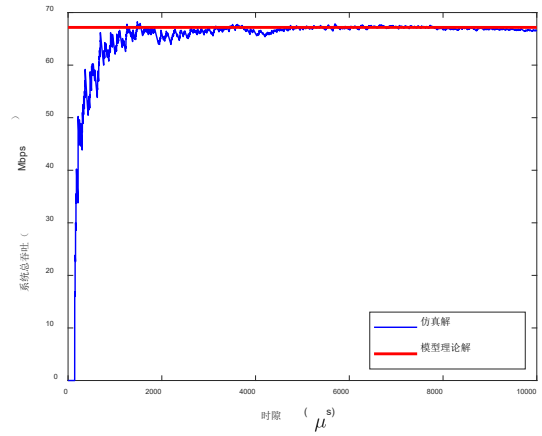
在仿真中，我们设置时隙长度为 1 微秒，最大仿真时长为 10 秒。在仿真初始时刻，两个 AP 同时进入可用信道评估状态，仿真结果如图 5-6 所示。

为验证模型预测精度，本题中以仿真值与理论值的均方根误差（RMSE，Root Mean Squared Error）作为精度误差指标，RMSE 的计算公式如下：

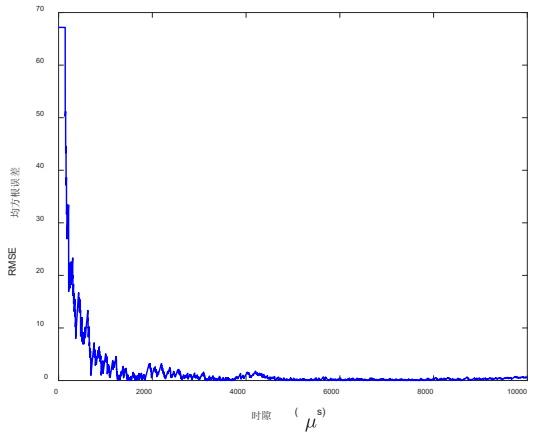
$$y_{rmse}=\sqrt{\frac{\sum_{i=1}^{N}(y_{pred}-y_{real})}{N}} \tag{5-25}$$

从图 5-6(a)(b)可以看出，仿真解在 2000 时隙后基本收敛到稳态，收敛后的吞吐量 $S_{sim}=66.8309\,\text{Mbps}$,略低于模型吞吐量 $S=67.1744\,\text{Mbps}$，两者均方根误差大致为 0.0367，差异较小，模型推导出的理论系统吞吐的精确度较高。图 5-6(c)(d)展示了马尔可夫模型中节点发送数据帧概率τ和碰撞概率p的拟合情况，其拟合误差分别为 2.21%和 1.66%，所提模型能较为准确的反应系统的实际情况。图 5-6(e)随机统计了 1000 个长度为 10 毫秒的仿

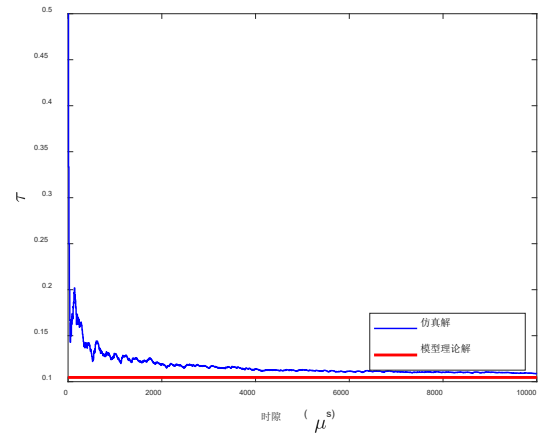
真系统吞吐量S_{sim}、发送概率τ和碰撞概率p，蓝色点为仿真统计值，红色点为模型理论值，从中可以看出，统计值与理论值相差不大，该模型能够很好的反应问题 1 场景下的信道接入机制。图 5-6(f)统计了系统每个时隙吞吐量的概率密度分布直方图，从中可以看出系统吞吐量变化贴近正态分布，且集中于[65.5, 68.5]区间内，分布较为密集，与模型推导结果相符。



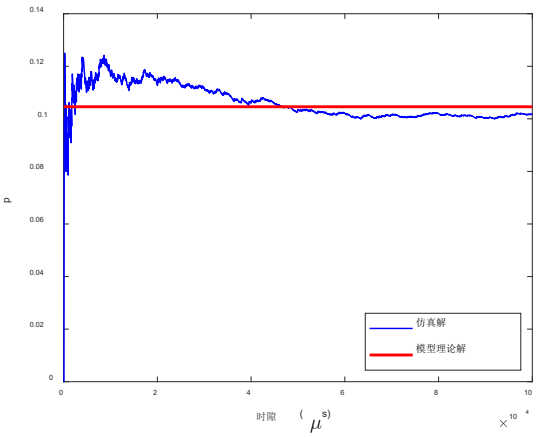
(a) 吞吐量结果对比



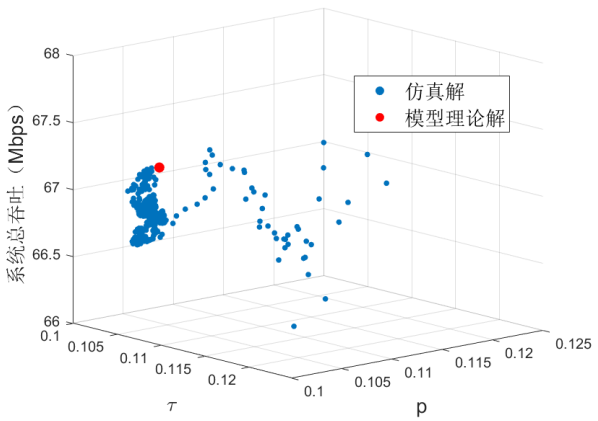
(b) RMSE 结果对比



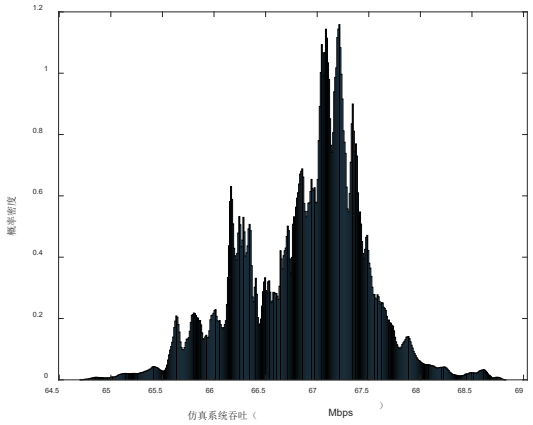
(c) 参数τ对比



(d) 参数p对比



(e) 参数p,τ和吞吐量对比



(f) 仿真吞吐概率密度

图 5-6 问题 1 仿真结果图

六、问题二的建模与求解

6.1 解题思路与建模

相较于问题 1，问题 2 将两个 AP 的物理层速率降低为原先的一半左右，使两个终端并发接收数据时 SIR 较高，信号能被成功解调，所以不存在碰撞导致的发送失败问题。但由于两个 BSS 仍然互听，所以需要考虑退避过程中的冻结机制，据此分析两个 AP 发送的时序过程，建立虚拟时隙分布模型，求解模型参数即可推导系统饱和吞吐量，最后与仿真结果进行对比并分析模型精度，问题 2 的整体解题思路和建模流程如图 6-1 所示。

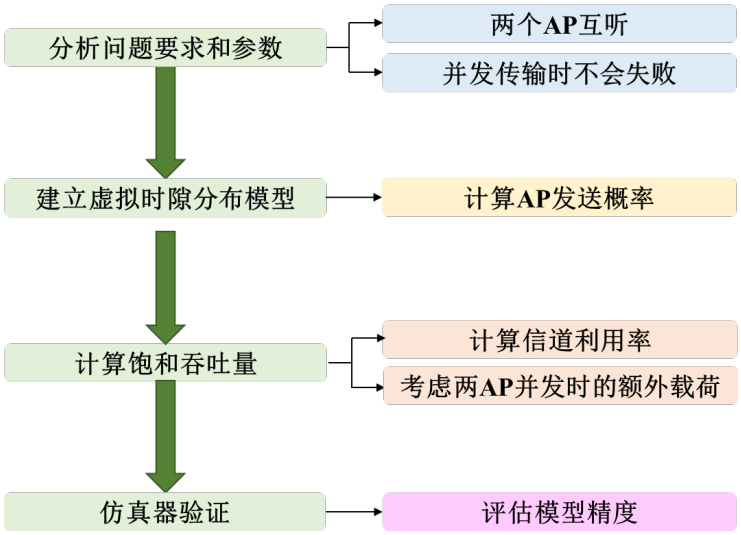


图 6-1 问题二的整体思路流程

6.2 数据传输碰撞概率分析

针对如图 6-2 所示的问题二中存在的两个 AP 互听但不存在同频干扰的场景，分析得到如图 6-3 所示的时隙流程图。

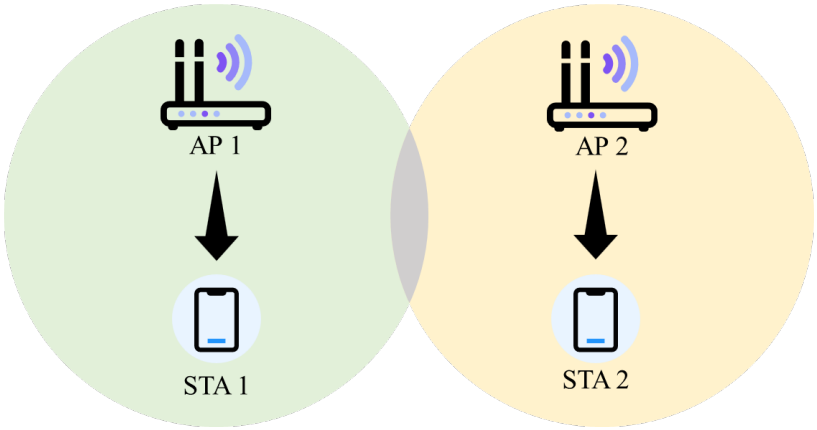


图 6-2 问题二场景图

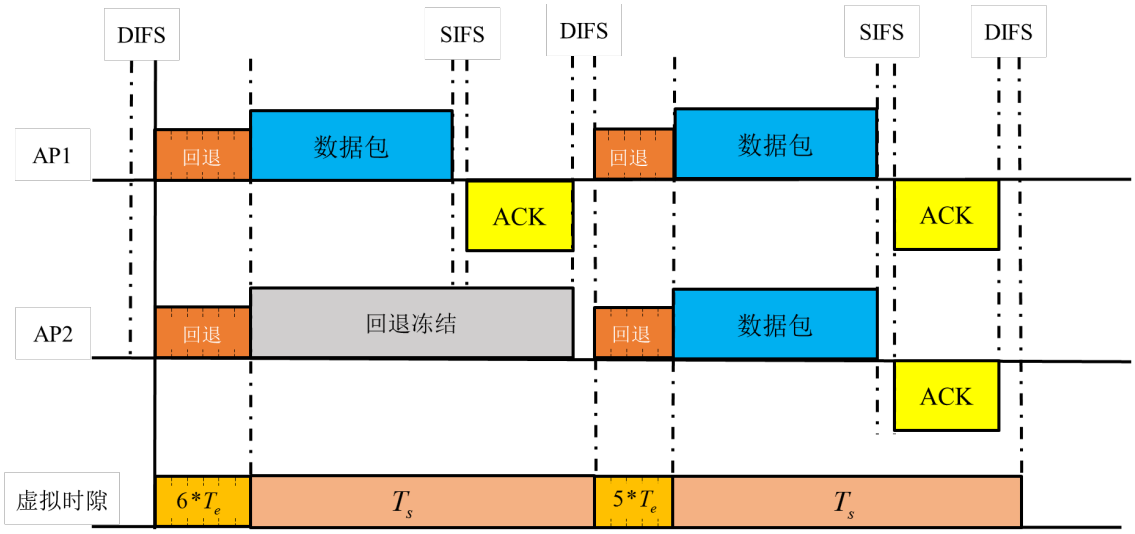


图 6-3 问题二时隙流程图

如图 6-3 所示，假设一个 AP 发送数据的概率为τ，空闲时隙的长度为T_e，空闲时隙的个数为N_e，由于不存在碰撞，所以在本题中 AP 仅可能处于空闲和发送两种虚拟时隙，在这些时隙中另一个 AP 也可能会发送数据，所以在一个完整的发送时间段内，信道上有数据在发送的的时隙数为$1+\tau$，空闲时隙个数的期望值为:

$$E[N_e]=\frac{1}{W_0}(0+1+...+W_0-1)=\frac{W_0-1}{2} \tag{6-1}$$

而在一个完整的发送流程中，有：

$$\tau=\frac{\tau+1}{E[N_e]+(\tau+1)} \tag{6-2}$$

联立式(6-1)和式(6-2)，根据附录 4 中W_0=16，可解得:

$$\tau=0.1310$$

6.3 吞吐量性能分析

与问题 1 不同，在本题中，由于不存在发送失败的概率，所以在计算吞吐量时不需要考虑发送失败的情况以及碰撞的时隙长度，但是相较于问题 1，本题需要多考虑两个 AP 同时发送时，有效数据载荷需要重复计算的问题。

基于上述分析，可得至少有一个 AP 发送数据的概率P_{tr}为：

$$P_{tr}=1-(1-\tau)^2 \tag{6-3}$$

其中，$(1-\tau)$代表其中某 1 个节点没有发送数据的概率，则$(1-\tau)^2$即为两个节点都不发送数据的概率，所以其互斥事件概率$1-(1-\tau)^2$即为至少有一个节点发送的概率。

由于本题中不存在节点发送数据失败的概率，所以在有节点发送(即$P_{tr}\neq 0$)数据的前

提条件下, 成功传输的概率为:

$$P_s = 0 \tag{6-4}$$

所以，饱和状态下吞吐量的表达式为：

$$S = \frac{(P_{tr} + \tau^2)E[P]v}{(1-P_{tr})\sigma + P_{tr}T_s} \tag{6-5}$$

其中，由于在部分发送时隙中，两个节点会同时发送数据，所以对于这一部分发送时隙，我们需要额外计算一份有效载荷 $\tau^2 E[P]v$，其余部分同问题 1。

故根据题目所给条件，联立(6-4),(6-6)令 $v = 275.3\,\text{Mbps}, T_M = \frac{30*8}{275.3} = 0.8718\mu\text{s},$ $E[P] = \frac{1500*8}{275.3} = 43.5888\,\mu\text{s}$，其余参数同问题 1 和附录 4，可解得:

$$S = 72.6290\text{Mbps}$$

6.4 仿真器性能验证

相较于问题 1，本题忽略了因碰撞而导致的传输失败，因此节点在进入数据传输阶段后必然发送成功，不会重传，但仍需考虑 AP 间互听而进入到冻结状态的问题，问题 2 仿真代码的流程如下图所示。

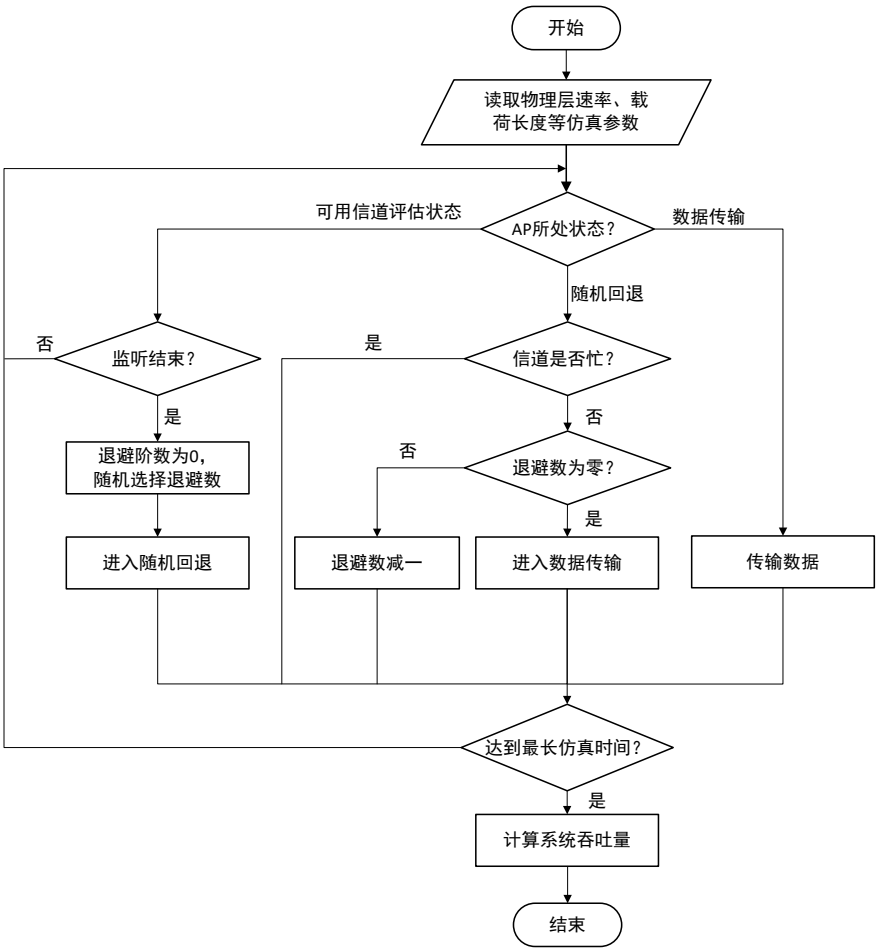
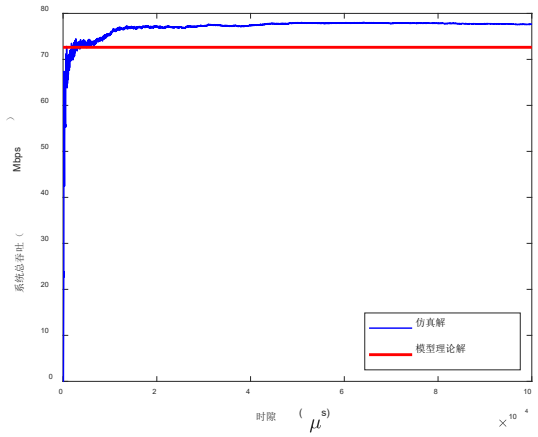


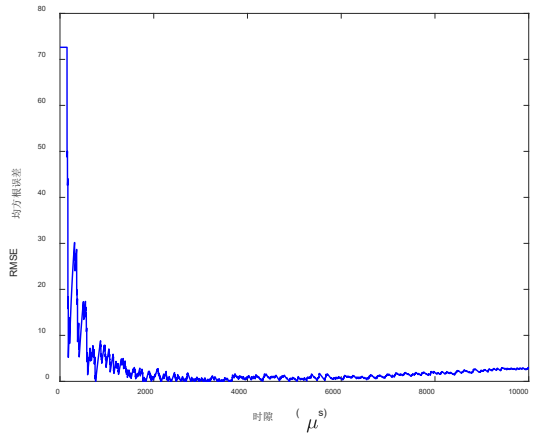
图 6-4 问题 2 仿真流程图

基于 Python 语言搭建的仿真器结果如图 6-5 所示，图 6-5(a)(b)给出了仿真过程中系统饱和吞吐量随时间的变化及仿真解与模型理论解的对比，从图中可以看出，随着时隙的增加，系统的饱和吞吐量趋于稳定，仿真收敛后的系统吞吐量 S_{sim}=77.7124Mbps，模型理论推导出的系统吞吐量 S=72.6290Mbps，平均 RMSE 为 5.0915。

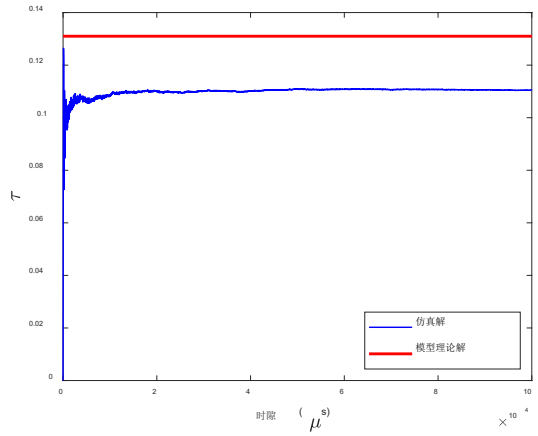
统计信道转化虚拟时隙数和传输成功次数可以获得节点发送数据帧概率τ随时间的变化曲线，如图 6-5(c)所示，仿真收敛后的$\tau = 0.1105$，小于模型理论解 0.1310。图 6-5(d)统计了系统每个时隙吞吐量的概率密度分布直方图，从中可以看出系统吞吐量变化贴近正态分布，且集中于[72, 81]区间内。



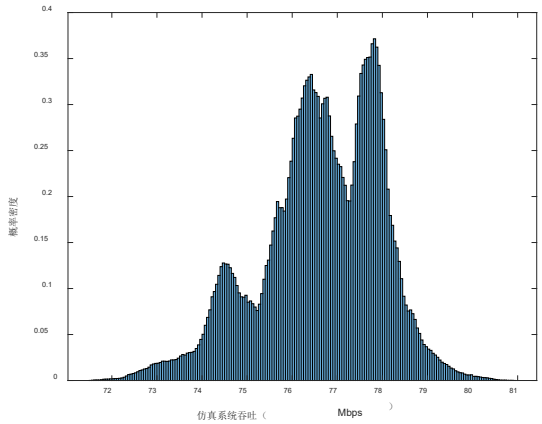
(a) 吞吐量结果对比



(b) RMSE 结果对比



(c) 参数τ对比



(d) 仿真吞吐概率密度

图 6-5 问题 2 仿真结果图

七、问题三的建模与求解

7.1 解题思路与建模

与问题 1、2 不同，本题假设在 2BSS 场景下，AP 间的 RSSI 为-90dBm，小于所规定的 CCA 门限(-72dBm)，二者之间不互听，互相无法感知到对方的存在，导致 AP 认为信道空闲，因此总是在退避和发送数据。并且考虑了无线通信环境的复杂多变性而导致的

丢包问题以及当两个 AP 发包交叠导致发包均失败的问题，即所谓的隐藏节点问题。本题的整体解题思路和建模流程如图 7-1 所示。

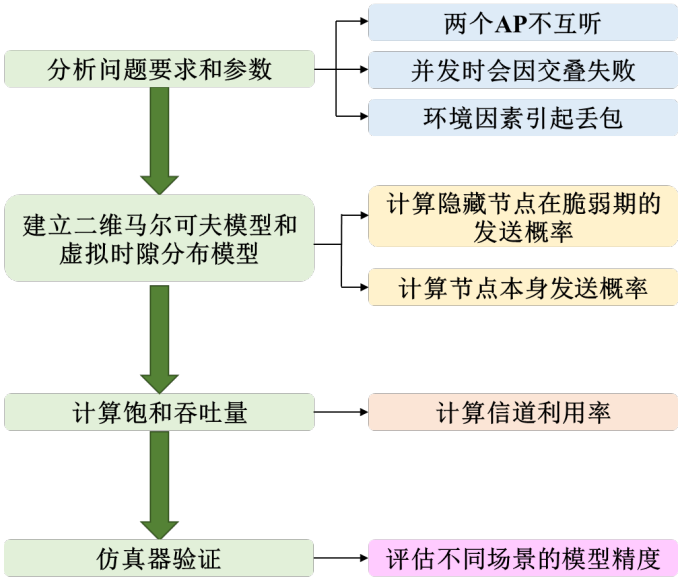


图 7-1 问题三的整体思路流程

7.2 数据传输碰撞概率分析

设发送失败重传的概率为 p_f，由于本题中考虑由于信道质量导致的丢包问题，所以即使两个 AP 发包在时间上没有交叠，也有可能因丢包而导致发送失败，据题目所给的丢包率 $P_e = 0.1$，发送失败的概率为 p_f 可表示为:

$$p_f = 0.9p + 0.1 \tag{7-1}$$

其中 p 为源 AP 发送数据包与隐藏 AP 数据包交叠的概率。

与问题 1、2 类似，根据 Bianchi 模型，竞争窗口的大小为：

$$W_i = \begin{cases} 2^i W_0, 0 \le i \le m \\ 2^m W_0, m \le i \le r \end{cases} \tag{7-2}$$

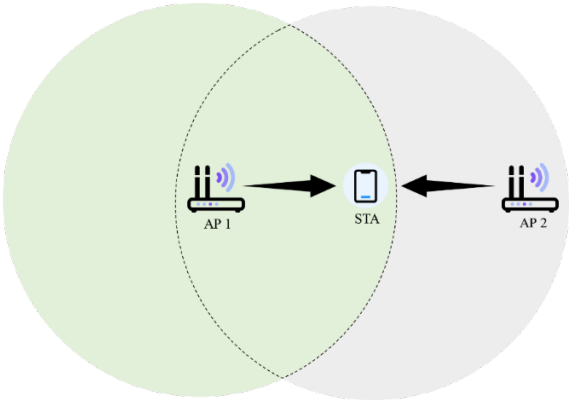


图 7-2 问题 3 场景图

将源 AP 状态节点$\{b(t),s(t)\}$表示为如图 7-3 所示的二维马尔可夫过程

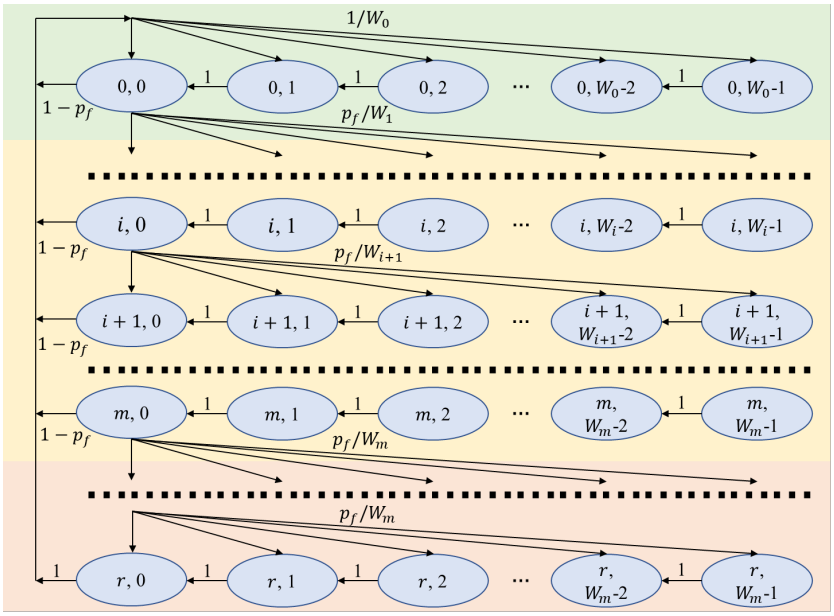


图 7.3 问题 3 马尔可夫过程

由图 7.2 并结合公式(7-1)可以得到以下几个一步转移概率:

$$P(i,k\mid i,k+1)=1,k\in[0,W_i-2],i\in[0,r] \tag{7-3}$$

$$P(0,k\mid i,0)=\frac{0.9-0.9p}{W_0},k\in[0,W_i-1],i\in[0,r] \tag{7-4}$$

$$P(i,k\mid i-1,0)=\frac{0.9p+0.1}{W_i},k\in[0,W_i-1],i\in[1,r] \tag{7-5}$$

$$P(0,k\mid r,0)=\frac{1}{W_0},k\in[0,W_0-1] \tag{7-6}$$

公式(7-3)表示了一次回退的情况，由于其不存在碰撞，所以概率为 1。

公式(7-4)表示当节点回退到 0 时，若发送成功（概率$1-p_f=0.9-0.9p$），则需要重新在W_0窗口内选择一个随机数（即某个随机数被选中的概率为$\frac{0.9-0.9p}{W_0}$），并转移到该状态。

公式(7-5)表示当在第$i-1$个回退阶段，节点回退到 0 时，若发送失败（即概率p_f），则需要首先调节窗口大小（指数为 2 增长），即W_{i-1}变为W_i。然后节点需要在更新后的窗口W_i中选择一个随机数（概率为$\frac{0.9p+0.1}{W_i}$），并转移至该状态。

公式(7-6)表示当节点到达最大的重传次数以后，无论成功还是失败，窗口都会重置，节点需要在重置后的窗口W_0中选择一个随机数（概率为$\frac{1}{W_0}$），并转移至该状态。

该二维马尔可夫模型稳态下的概率分布为:

$$b_{i,k}=\lim_{t\to\infty}P\{s(t)=i,b(t)=k\},i\in(0,m),k\in(0,W_i-1) \tag{7-7}$$

通过稳态分布的归一化条件，可得：

$$1=\sum_{k=0}^{W_i-1}\sum_{i=0}^{r}b_{i,k} \tag{7-8}$$

基于马尔可夫链的规则，我们可得:

$$b_{0,0}=\begin{cases}\dfrac{2(1-p_f)(1-2p_f)}{(1-2p_f)(1-{p_f}^{r+1})+W_0(1-p_f)(1-(2p_f)^{r+1})}, & r\le m\\ \dfrac{2(1-p_f)(1-2p_f)}{W_0(1-(2p_f)^{m+1})(1-p_f)+(1-2p_f)(1-{p_f}^{r+1})+W_0 2^m {p_f}^{m+1}(1-{p_f}^{r-m})(1-2p_f)}, & m<r\end{cases} \tag{7-9}$$

源节点在选择的时隙传输数据的概率τ_1为:

$$\tau_1=\sum_{i=0}^{r}b_{i,0}=\frac{1-{p_f}^{r+1}}{1-p_f}*b_{0,0} \tag{7-10}$$

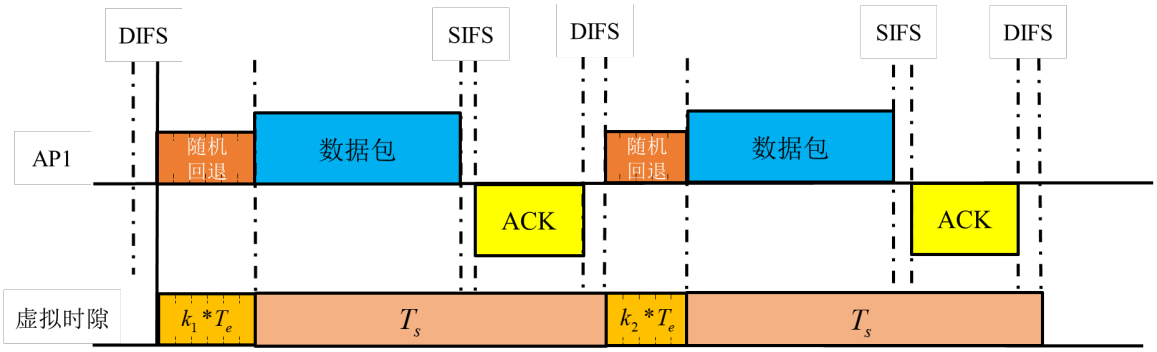


图 7-4 (a)在不考虑 AP2 时 AP1 的时隙流程图

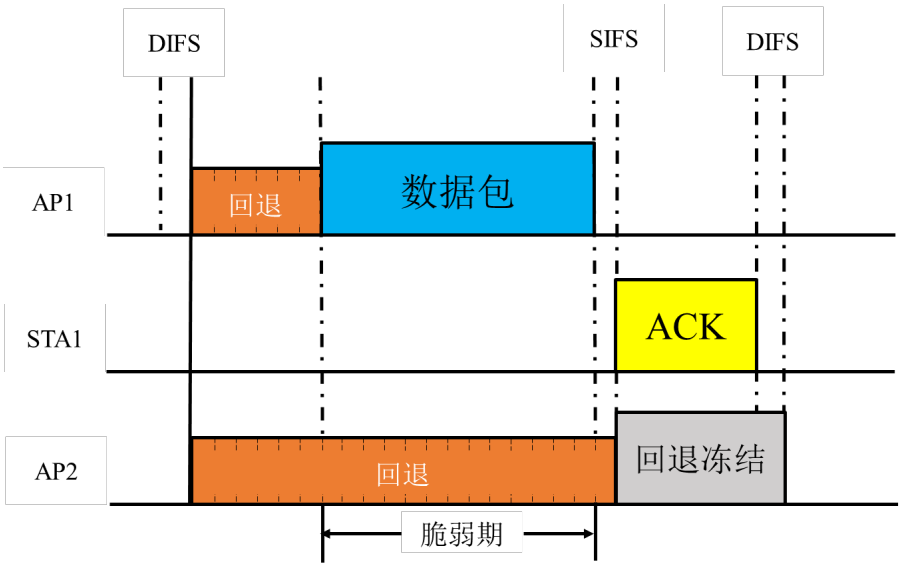


图 7-4 (b) 脆弱期示例图

设 AP1 和 AP2 可能发生碰撞的时期（即脆弱期，如图 7-4 所示）的长度为V (以回退时隙为单位，向下取整)， X 为竞争窗口大小大于V 的最小回退阶段，则隐藏节的发送概

率 τ_2 可表示为：

$$\tau_2=\sum_{i=0}^{r}\sum_{k=0}^{V}b_{i,k}=\begin{cases}\left[(V+1)*(\frac{1-p_f^{\,r+1}}{1-p_f})-(\frac{V(V+1)}{2W_0})*(\frac{1-(\frac{p_f}{2})^{r+1}}{1-\left(\frac{p_f}{2}\right)})*\right]*b_{0,0}, & V<W\\ \left[\frac{1}{2}*\left(\frac{1-p_f^{\,X}}{1-p_f}\right)+\frac{W_0}{2}*\left(\frac{1-(2p_f)^X}{1-(2p_f)}\right)+(V+1)*\left(\frac{p_f^{\,X}-p_f^{\,r+1}}{1-p_f}\right)-\left(\frac{V(V+1)}{2W_0}\right)*\left(\frac{\left(\frac{p_f}{2}\right)^X-\left(\frac{p_f}{2}\right)^{r+1}}{1-\left(\frac{p_f}{2}\right)}\right)\right]*b_{0,0}, & W_{X-1}<V<W_X\\ & 1\le X\le r\\ 1 & V>W_r\end{cases} \tag{7-11}$$

在本问题的场景中，碰撞概率 p 即为源节点和隐藏节点的碰撞概率，可表示为：

$$p=1-(1-\tau_2)=\tau_2 \tag{7-12}$$

联立（7-1）、（7-8）、（7-9）、（7-10）、(7-11)，并参照题目及附录 6 中所提供的参数，可分别解得不同参数下的 p_f 、 τ_1 、 τ_2 结果如表 7-1 所示。

7.3 吞吐量性能分析

与问题 1、2 类似，根据 Bianchi 提出的模型，饱和状态下吞吐量的表达式为：

$$S=\frac{E[t]}{E[ts]}*v \tag{7-13}$$

其中分子部分表示一个时隙内有效载荷传输的平均时间，分母部分表示一个时隙的平均时间。为了计算吞吐量 S，首先我们需要分析在一个时隙的时间内可能发生的情况。由于在本题仅考虑一个隐藏节点和源节点在竞争信道，而源节点无法监听到隐藏节点，会一直认为信道空闲，所以至少有一个 AP 发送数据的概率 P_{tr} 为：

$$P_{tr}=1-\left(1-\tau_1\right)^2 \tag{7-14}$$

其中，$(1-\tau_1)$ 代表其中一个 AP 没有发送数据的概率，则 $\left(1-\tau_1\right)^2$ 即为两个 AP 都不发送数据的概率，所以其互斥事件的概率 $1-\left(1-\tau_1\right)^2$ 即为至少有一个节点发送的概率。

在本场景中，由于仅考虑一个源节点，一个隐藏节点，若两个节点在发包时间上有交叠，则会导致传输的失败。所以，仅当源节点发送数据包，而隐藏节点不发送数据包时，才能够成功传输。所以在有节点发送(即 $P_{tr}\neq 0$)的前提条件下, 成功传输的概率为:

$$P_s=\frac{2\tau_1(1-\tau_2)}{P_{tr}}=\frac{2\tau_1(1-\tau_2)}{1-(1-\tau_1)^2} \tag{7-15}$$

其中分子中的 τ_1 代表源节点正在发送，$(1-\tau_2)$ 代表隐藏节点没有发送，而成功发送仅需场景中的两个节点中的其中一个发送即可，故将分子再乘以 2。

所以，与问题 1 类似，饱和状态下吞吐量的表达式为：

$$S=\frac{P_s P_{tr} E[P] v}{(1-P_{tr})\sigma + P_{tr} P_s T_s + P_{tr}(1-P_s) T_c} \tag{7-16}$$

其中，发送时隙长T_s和发生碰撞时隙长T_c计算方式同问题 1，由于题目中所给 AP 发送包的载荷长度为定值，所以在本题中$E[P]=E^*[P]$，而数据帧头传输时间H计算方式也与问题 1 相同。

参照题目及附录中所提供的参数可解得系统吞吐量S如表 7-1 所示:

表 7-1 p_f、τ_1、τ_2、S在 7 种场景下的不同解

	场景1	场景2	场景3	场景4	场景5	场景6	场景7
CW_{min}	16	16	32	16	16	32	16
CW_{max}	1024	1024	1024	1024	1024	1024	1024
最大重传次数r	32	6	5	32	6	5	32
物理层速率v (Mbps)	455.8	286.8	286.8	286.8	158.4M	158.4	158.4
p_f	0.2291	0.2558	0.1963	0.2502	0.2770	0.2343	0.2961
τ_1	0.1176	0.1176	0.0606	0.1176	0.1176	0.0606	0.1176
τ_2	0.2291	0.2558	0.1963	0.2502	0.2770	0.2343	0.2961
系统吞吐量S (Mbps)	52.7934	46.4550	40.8410	46.8293	37.9241	33.5274	38.3632

7.4 仿真器性能验证

问题 3 进一步考虑了丢包和隐藏节点的因素。具体而言，在数据传输阶段，有 10%的概率发生丢包，从而导致传输失败，进而返回随机回退阶段或丢弃数据帧。此外，由于 AP1 与 AP2 互为隐藏节点，它们无法相互监听彼此的信号。因此，AP 只能根据广播的 ACK 确认帧来判断同频干扰的存在性，这进一步增加了由数据包交叠导致的传输失败概率。问题 3 仿真代码的流程如下图所示。

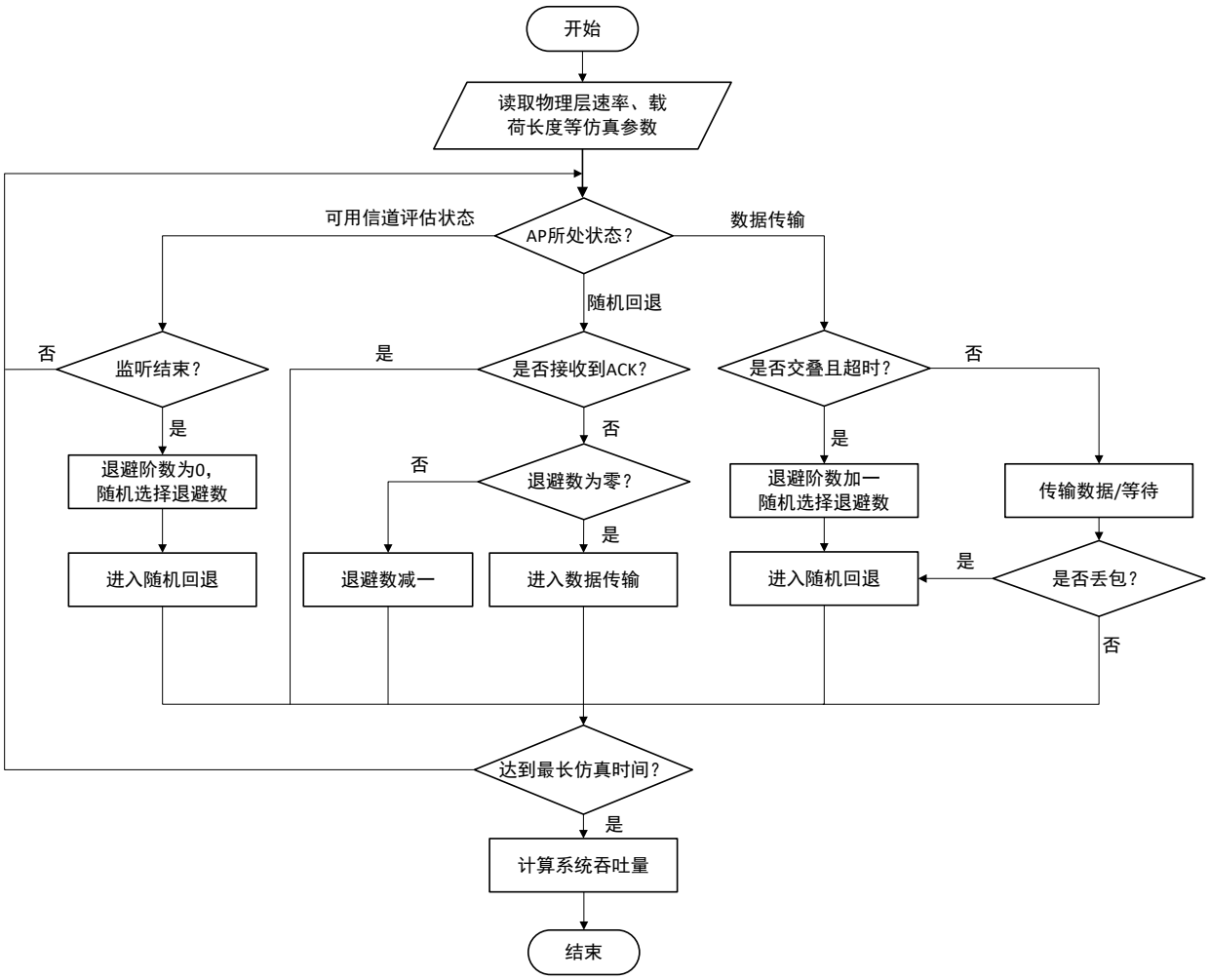
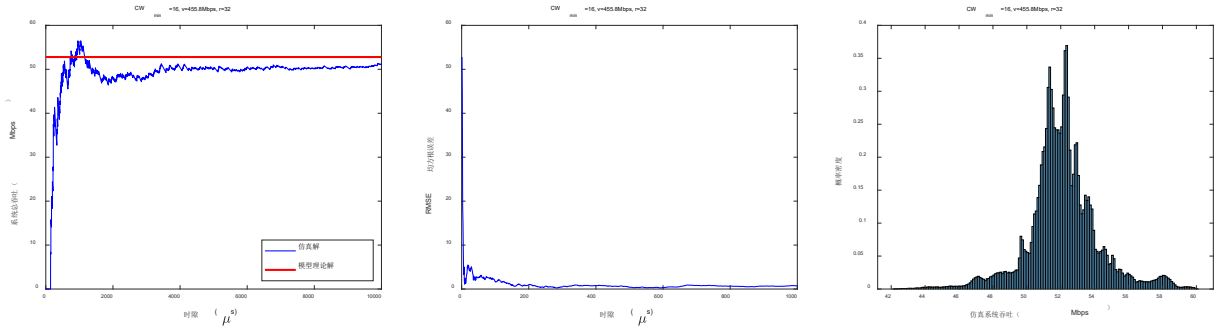


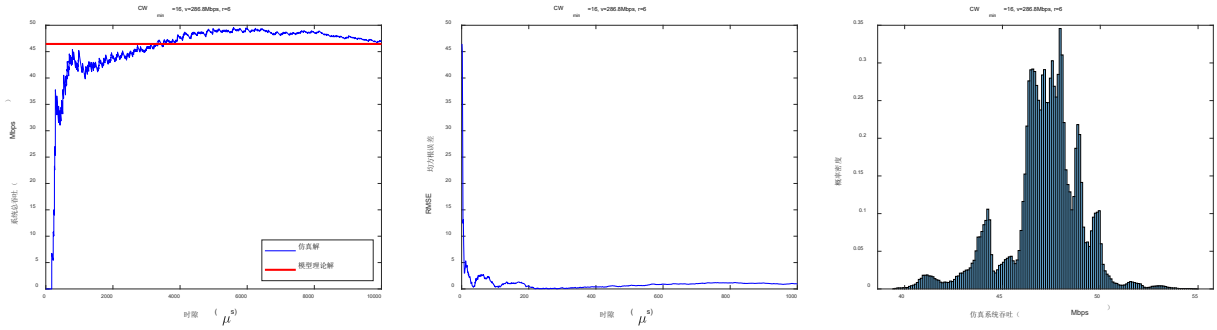
图 7-5 问题 3 仿真流程图

为验证问题 3 所提模型的准确性，参照附录 4（1 组参数）和附录 6（6 组参数）所实现的仿真结果如图 7-6 至图 7-12 所示，从图(a)(b)中可以看出，在不同参数下，仿真解的吞吐量能迅速收敛到模型理论解附近，同时 RMSE 迅速收敛到 0，7 组实验结果的最大误差为 3.685%，从图(c)中可以看出，每时隙统计的仿真系统吞吐量概率分布大致符合正态分布曲线，且模型理论解大致位于曲线中心。



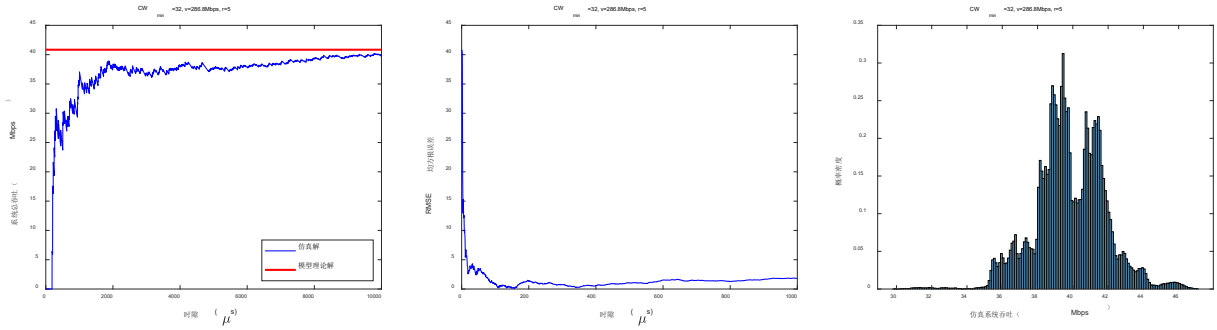
(a) 吞吐量结果对比　　(b) RMSE 结果对比　　(c) 仿真吞吐概率密度

图 7-6 场景 1 $CW_{min}=16$, $v=455.8$Mbps, $r=32$



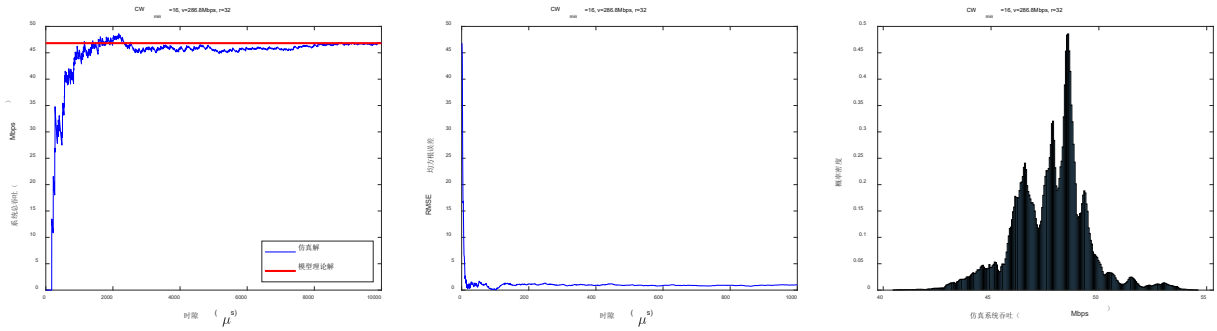
(a) 吞吐量结果对比　　(b) RMSE 结果对比　　(c) 仿真吞吐概率密度

图 7-7 场景 2 $CW_{min}=16,\ v=286.8\text{Mbps},\ r=6$



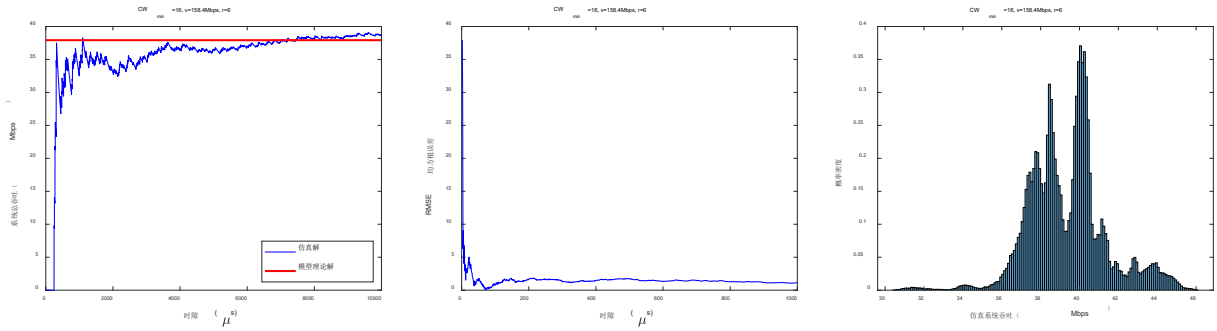
(a) 吞吐量结果对比　　(b) RMSE 结果对比　　(a) 仿真吞吐概率密度

图 7-8 场景 3 $CW_{min}=32,\ v=286.8\text{Mbps},\ r=5$



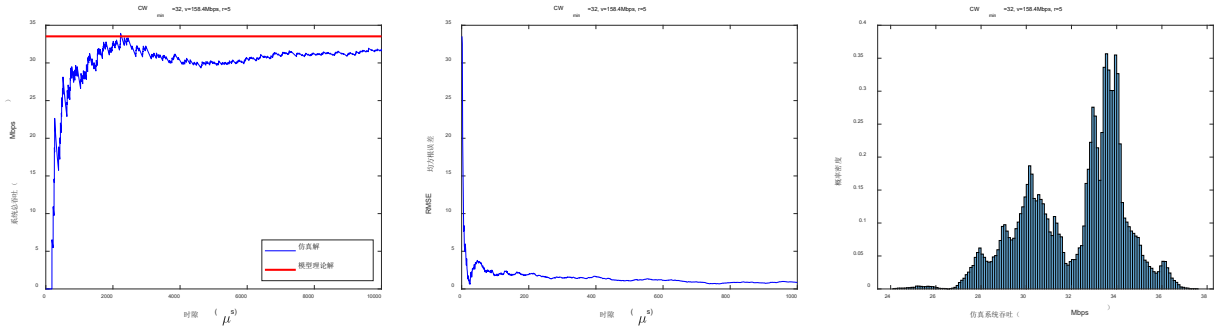
(a) 吞吐量结果对比　　(b) RMSE 结果对比　　(a) 仿真吞吐概率密度

图 7-9 场景 4 $CW_{min}=16,\ v=286.8\text{Mbps},\ r=32$



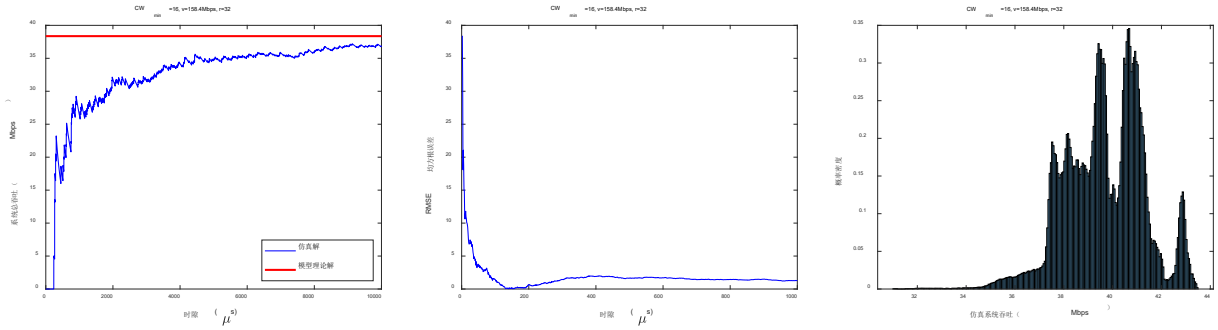
(a) 吞吐量结果对比　　(b) RMSE 结果对比　　(a) 仿真吞吐概率密度

图 7-10 场景 5 $CW_{min}=16,\ v=158.4\text{Mbps},\ r=6$



(a) 吞吐量结果对比　　(b) RMSE 结果对比　　(a) 仿真吞吐概率密度

图 7-11 场景 6 $CW_{min}=32$, $v=158.4$Mbps, $r=5$



(a) 吞吐量结果对比　　(b) RMSE 结果对比　　(c) 仿真吞吐概率密度

图 7-12 场景 7 $CW_{min}=16$, $v=158.4$Mbps, $r=32$

表 7-3 结合具体参数给出了 7 种场景下模型理论吞吐量与仿真吞吐量的对比，经仿真验证，模型均方根误差小于 2，具有较高的准确性。

表 7-3 7 种场景下的仿真和模型对比

	场景1	场景2	场景3	场景4	场景5	场景6	场景7
CW_{min}	16	16	32	16	16	32	16
CW_{max}	1024	1024	1024	1024	1024	1024	1024
最大重传次数r	32	6	5	32	6	5	32
物理层速率v(Mbps)	455.8	286.8	286.8	286.8	158.4	158.4	158.4
模型吞吐量(Mbps)	52.7934	46.4550	40.8410	46.8293	37.9241	33.5274	38.3632
仿真吞吐量(Mbps)	51.7832	47.0085	39.3896	47.4277	38.9486	32.0260	38.9473
RMSE 均方根误差	0.6800	0.9790	1.7946	0.9554	1.1182	0.8889	1.3204

八、问题四的建模与求解

8.1 解题思路与建模

与前面 3 个问题不同，本题考虑的是一个 3BSS 系统，且这个系统中的节点部分互听，其中 AP1 与 AP2 之间，AP2 与 AP3 之间 RSSI 均为-70dBm，所以 AP2 与两者都互听；而 AP1 与 AP3 之间 RSSI 为-96dBm，所以 AP1 与 AP3 不互听。所以需要设计两种不同转移概率的二维马尔可夫链，并据此得到 AP1 和 AP3 的发送概率与碰撞概率之间的关系，进而计算信道利用率得出系统吞吐量。本题的整体解题思路和建模流程如图 8-1 所示。

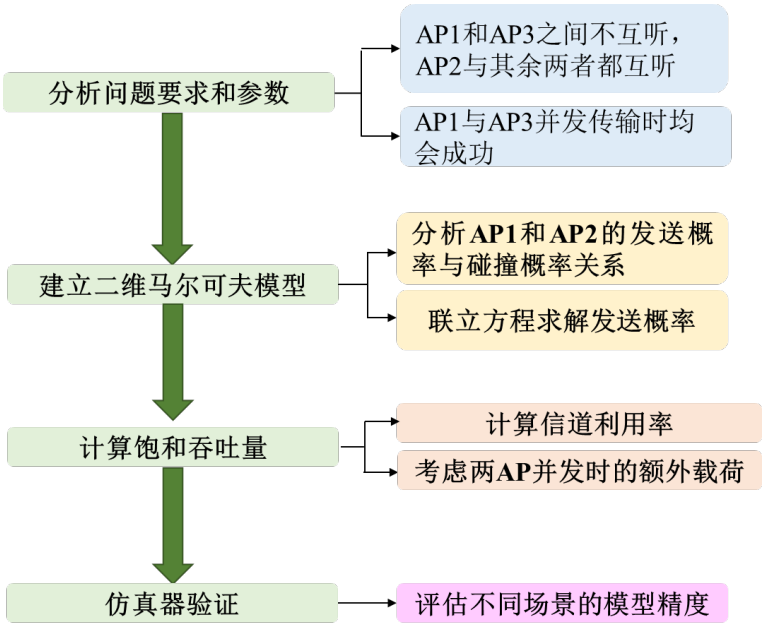


图 8-1 问题 4 的整体思路流程

8.2 三个 AP 的数据包发送概率与碰撞概率分析

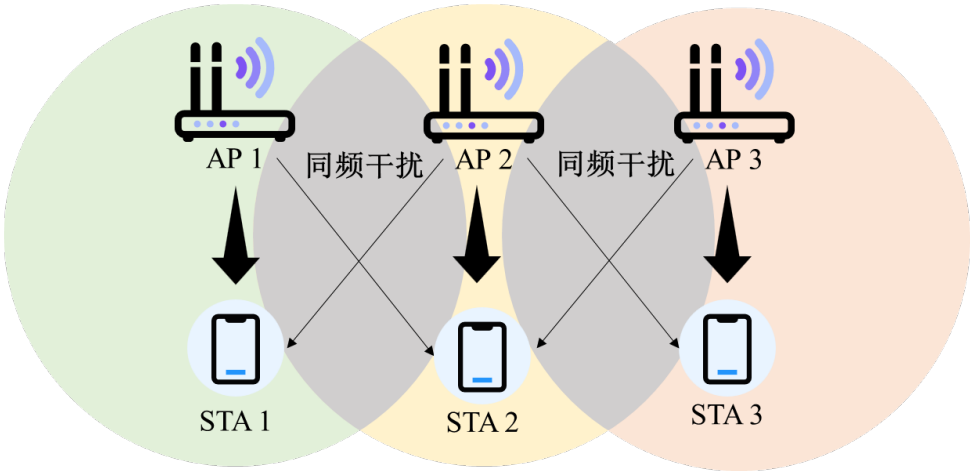


图 8-2 3BSS 场景示意图

针对问题 4 存在的 3BSS 场景，如图 8-2 所示，由于 AP1 与 AP2 之间、AP2 与 AP3 之间 RSSI 较高，AP1 与 AP3 之间 RSSI 较低，所以 AP1 与 AP3 不互听、AP2 与两者都互听，且如图 8-3 所示，AP1 和 AP3 发包时间交叠时，由于 SIR 较大，两者发送均会成功。所以对于 AP1 来说，其传输的数据仅在与 AP2 传输的数据发生碰撞时导致失败重传，而 AP3 是否传输数据不会影响其传输。所以对于 AP1 来说，可以考虑仅包含 AP1 和 AP2 的 2BSS 系统来计算其在每一个虚拟时隙的发送概率和碰撞概率，AP3 与 AP1 的情况相同，所以它们有相同的发送概率和碰撞概率。而对于 AP2 来说，在它进行数据传输时，只要 AP1 和 AP3 任意一个发送数据，都会导致数据传输失败。所以在计算 AP2 的发送概率和碰撞概率时必须考虑 3BSS 系统的中的概率关系。

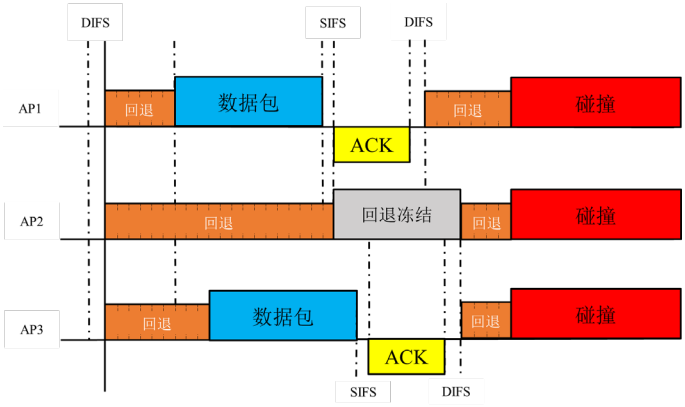


图 8-3 时序流程示例图

假设 AP1 和 AP3 在某一个虚拟时隙的发送概率为τ_1、发生碰撞的条件概率为p_1，假设 AP2 在每一个虚拟时隙的发送概率为τ_2，发生碰撞的条件概率为p_2，下面将对这四个变量进行求解。

首先考虑 AP1 和 AP2 组成的 2BSS 系统中，两个 AP 同时回退到 0 而同时发送数据时产生同频干扰而导致数据传输失败的场景，根据 Bianchi 模型，将 AP1 的节点转移过程$\{b(t),s(t)\}$表示为如图 8-4 所示的二维马尔可夫过程：

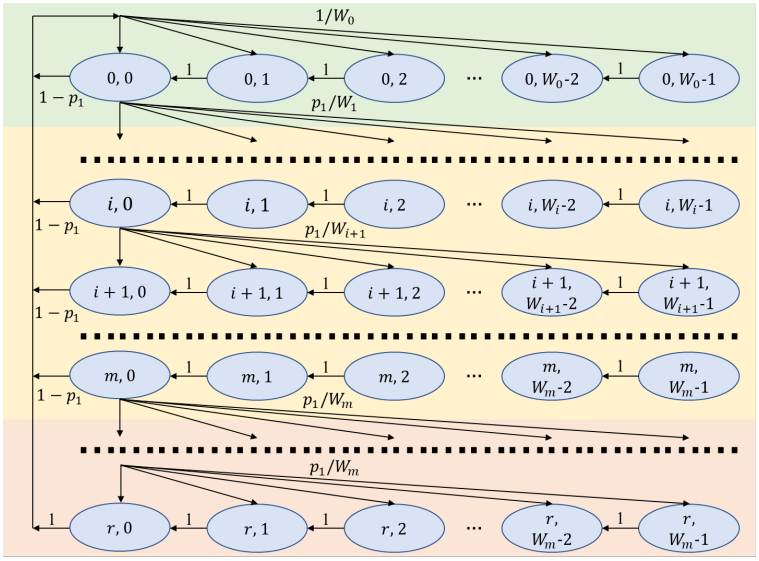


图 8-4 AP1 马尔可夫过程

由上图所示的马尔可夫过程，并通过与问题 1 相同的推导过程，可以得出 AP1 在一个时隙发送数据帧的概率为：

$$\begin{aligned}\tau_1 &= \sum_{0}^{r} b_{i,0} \\ &= b_{0,0} * \frac{1-p_1^{r+1}}{1-p_1} \\ &= \begin{cases} \dfrac{2(1-2p_1)(1-2p_1^{r+1})}{(1-2p_1)(1-p_1^{r+1})+W_0(1-p_1)(1-(2p_1)^{r+1})}, & r \le m \\ \dfrac{2(1-2p_1)(1-2p_1^{r+1})}{W_0(1-(2p_1)^{m+1})(1-p_1)+(1-2p_1)(1-p_1^{r+1})+W_0 2^m p_1^{m+1}(1-p_1^{r-m})(1-2p_1)}, & m < r \end{cases}\end{aligned} \tag{8-1}$$

同理可得 AP1 的发送概率：

$$\tau_2 = \begin{cases} \dfrac{2(1-2p_2)(1-2p_2^{r+1})}{(1-2p_2)(1-p_2^{r+1})+W_0(1-p_2)(1-(2p_2)^{r+1})}, & r \le m \\ \dfrac{2(1-2p_2)(1-2p_2^{r+1})}{W_0(1-(2p_2)^{m+1})(1-p_2)+(1-2p_2)(1-p_2^{r+1})+W_0 2^m p_2^{m+1}(1-p_2^{r-m})(1-2p_2)}, & m < r \end{cases} \tag{8-2}$$

对于 AP1 来说，当其发送数据时，其数据包发生碰撞即为 AP2 发送数据的概率，即:

$$p_1 = \tau_2 \tag{8-3}$$

而对于 AP2 来说，当其发送数据时，其数据包发生碰撞即为 AP1 和 AP2 有一个发送数据或者同时发送数据的概率，即：

$$p_2 = 1-(1-\tau_1)^2 \tag{8-4}$$

在给定 r 、 m 和 W_0 的条件下，联立式(8-1)、(8-2)、(8-3)和(8-4)这四个方程即可解出 τ_1、 τ_2、 p_1 和 p_2 这四个变量。

据题意以及附录 6 提供的不同 r 、 m 和 W_0 参数可求出以下解，如表 8-1 所示：

表 8-1 AP1 和 AP2 的发送概率与碰撞概率在不同参数下的解

r	32	6	5	32
m	6	6	5	5
W_0	16	16	32	16
τ_1	0.1067	0.1067	0.0573	0.1067
τ_2	0.0892	0.0893	0.0532	0.0892
p_1	0.0892	0.0893	0.0532	0.0892
p_2	0.2021	0.2021	0.1113	0.2021

8.3 吞吐量性能分析

根据 Bianchi 提出的模型，饱和状态下吞吐量的表达式为：

$$S = \frac{E[t]}{E[ts]} * v \tag{8-5}$$

其中分子部分表示一个时隙内有效载荷传输的平均时间，分母部分表示一个时隙的平均时间。

为了计算吞吐量S，首先我们需要分析在一个时隙的时间内可能发生的情况。

由于 AP1 和 AP3 发包时间交叠时，SIR 较大，两者发送均成功，即他们之间是否同时发送数据包不会影响各自发送包的成功概率，所以先考虑 AP1 和 AP2 组成 2BSS 系统。在这个系统中，AP1 在每个虚拟时隙发送数据包的概率为τ_1，AP2 发送数据包的概率为τ_2，令P_{tr}为在考虑的时隙内至少有一个 AP 发送数据的概率，则在这个 2BSS 系统中有：

$$P_{tr} = 1-(1-\tau_1)(1-\tau_2) \tag{8-6}$$

其中，$(1-\tau_1)$代表 AP1 节点没有发送数据的概率，$(1-\tau_2)$代表 AP2 节点没有发送数据的概率，则$(1-\tau_1)(1-\tau_2)$即为两个节点都不发送数据的概率，所以其互斥事件的概率，也就是$1-(1-\tau_1)(1-\tau_2)$，即为至少有一个节点发送的概率。

而在这个 2BSS 系统中，AP1 成功传输的概率为:

$$P_{s,1} = \frac{\tau_1(1-\tau_2)}{P_{tr}} = \frac{\tau_1(1-\tau_2)}{1-(1-\tau_1)(1-\tau_2)} \tag{8-7}$$

AP2 成功传输的概率:

$$P_{s,2} = \frac{\tau_2(1-\tau_1)}{P_{tr}} = \frac{\tau_2(1-\tau_1)}{1-(1-\tau_1)(1-\tau_2)} \tag{8-8}$$

而在 3BSS 系统中，相对上述考虑的 2BSS 系统，其增加了一项 AP3 的成功概率，而其与 AP1 相同，所以总的成功概率和：

$$P_s = 2P_{s,1} + P_{s,2} \tag{8-9}$$

所以，问题 4 中所考虑的饱和状态下吞吐量的表达式为：

$$S = \frac{P_s P_{tr} E[P] v}{(1-P_{tr})\sigma + P_{tr}P_s T_s + P_{tr}(1-P_s)T_c} \tag{8-10}$$

其中，发送时隙长T_s和发生碰撞时隙长T_c计算方式同问题 1，由于题目中所给 AP 发送包的载荷长度为定值，所以在本题中$E[P] = E^*[P]$，而数据帧头传输时间H计算方式也与问题 1 相同。

据题目所给条件，并参照附录 4 和附录 6 中所提供的参数可解得如表 8-2 所示的系统吞吐量：

表 8-2 根据题目及附录参数计算出的吞吐量

	场景1	场景2	场景3	场景4	场景5	场景6	场景7
CW_{min}	16	16	32	16	16	32	16
CW_{max}	1024	1024	1024	1024	1024	1024	1024
最大重传次数r	32	6	5	32	6	5	32
物理层速率v（Mbps）	455.8	286.8	286.8	286.8	158.4M	158.4	158.4
系统吞吐量S (Mbps)	108.4228	98.7974	82.8179	98.814	82.7612	71.2892	82.7761

8.4 仿真器性能验证

问题 4 中考虑了 3 个 BSS 的场景，其中 AP2 与 AP1、AP3 互听且存在同频干扰，AP1 与 AP3 之间不互听且不存在同频干扰，因此需要在监听信道是否忙时对正在传输的 AP 进行分类讨论，仿真器的具体伪代码如表 8-3 所示。

表 8-3 问题 4 仿真器伪代码

问题 4：3BSS 场景下仿真器伪代码

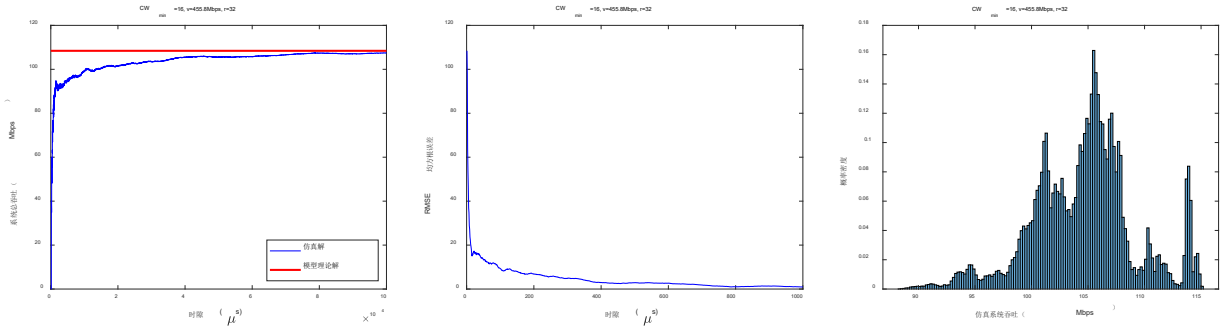
输入：载荷长度l，PHY 头时长pl，物理层速率v，竞争窗口初始长度$\mathrm{CW_{min}}$，最大重传次数r，最大传输时间T

输出：饱和状态下的吞吐量S

初始化信道状态C为空闲，数据传输总量$Data = 0$
for t=1 **to** T **do**：
  for i=1 to 3 do: \\ 3 个 AP
    if AP(i)信道监听结束：
      判断此刻 SIR 是否大于 CCA 阈值并记录
    end if
    if AP(i)状态为可用信道评估且监听结束：
      随机从[0,$\mathrm{CW_{min}} - 1$]选取回退数并乘以退避时长
      进入随机回退状态
    else if AP(i)的状态为随机回退：
      if 回退数可以整除退避时长且监听到信道状态为忙：
        冻结节点直到信道状态为闲
      end if
      回退数减一
      if 回退数为 0：
        进入数据传输阶段
      end if
    else： \\ 处于数据传输
    更新信道状态为正在传输数据
    if 信号传输 SIR 大于阈值，能够成功解调

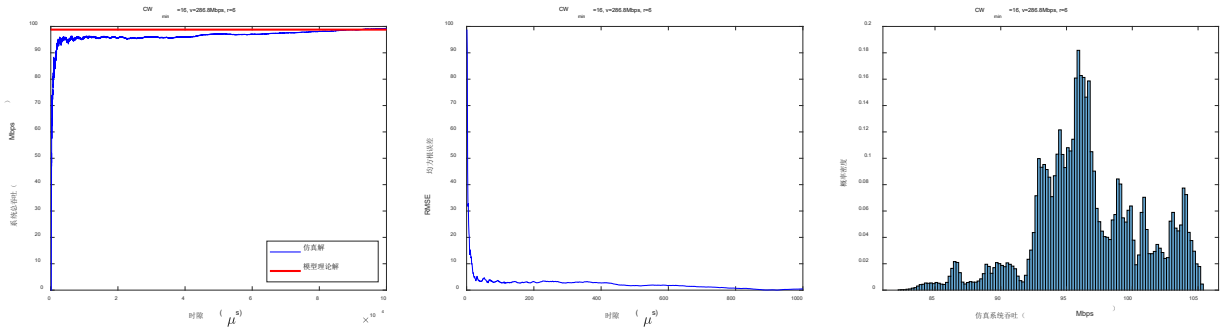
```
21        if 传输时间大于成功传输时长
22          传输成功
23        end if
24      else if
25        传输失败
26      end if
27    end for
28    信道状态更新
29 end for
30 统计发送成功的数据量，计算系统饱和吞吐量S
```

图 8-5 到图 8-11 为参照附录 4（1 组参数）和附录 6（6 组参数）所运行的仿真结果，从中可以看出，在场景 1，场景 2，场景 4 中仿真解能够很快收敛到模型理论解附近，模型精度较高，但在其余场景中，模型的均方根误差相对较大，其概率密度曲线也较为平缓。



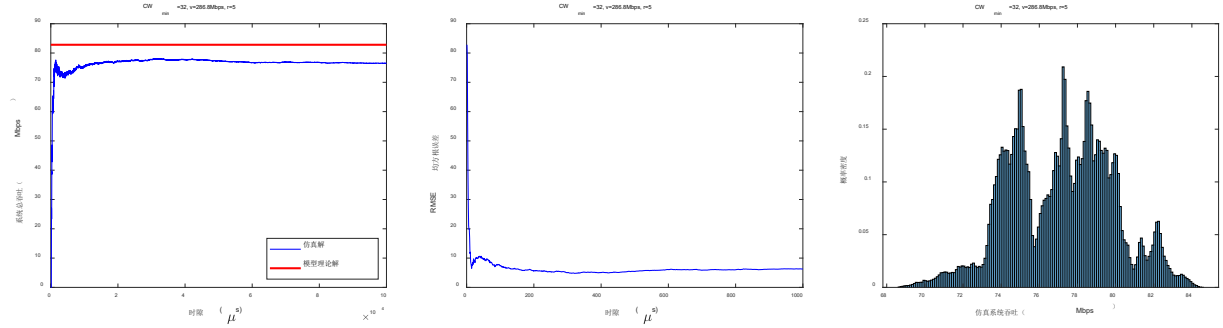
(a) 吞吐量结果对比　　(b) RMSE 结果对比　　(c) 仿真吞吐概率密度

图 8-5 场景 1 $CW_{min}=16,\ v=455.8\text{Mbps},\ r=32$



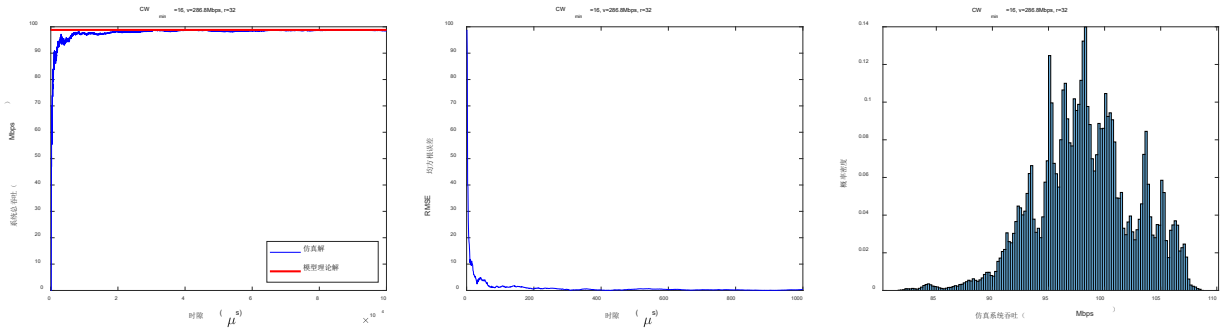
(a) 吞吐量结果对比　　(b) RMSE 结果对比　　(c) 仿真吞吐概率密度

图 8-6 场景 2　$CW_{min}=16,\ v=286.8\text{Mbps},\ r=6$



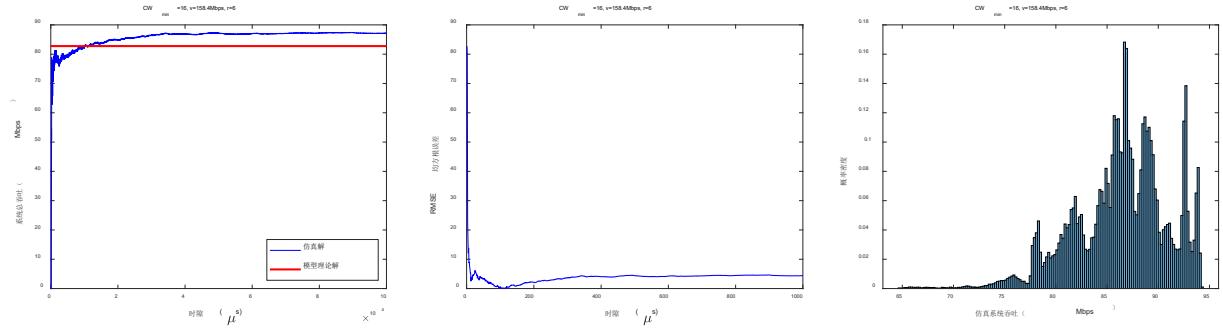
(a) 吞吐量结果对比　　(b) RMSE 结果对比　　(c) 仿真吞吐概率密度

图 8-7 场景 3　$CW_{min}=32,\ v=286.8\text{Mbps},\ r=5$



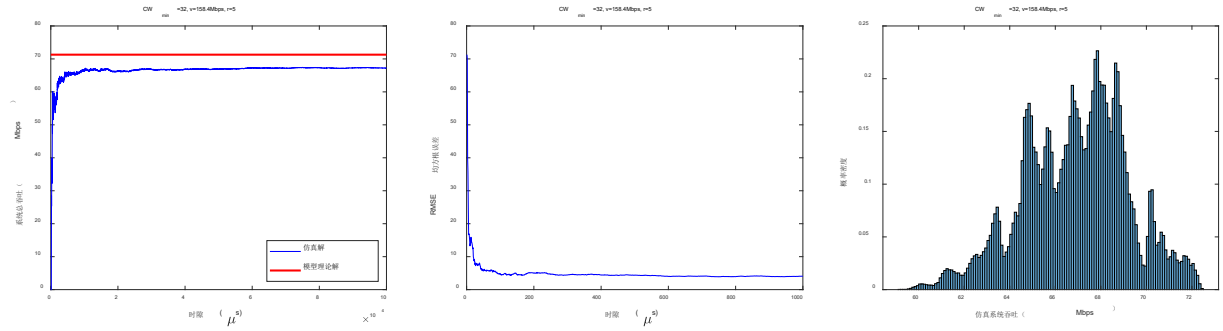
(a) 吞吐量结果对比　　(b) RMSE 结果对比　　(c) 仿真吞吐概率密度

图 8-8 场景 4 $CW_{min}=16$, $v=286.8$Mbps, $r=32$



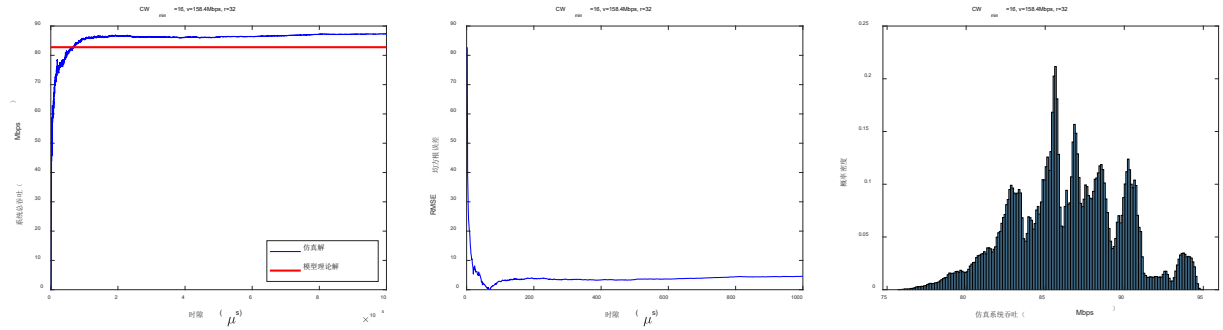
(a) 吞吐量结果对比　　(b) RMSE 结果对比　　(c) 仿真吞吐概率密度

图 8-9 场景 5 $CW_{min}=16$, $v=158.4$Mbps, $r=6$



(a) 吞吐量结果对比　　(b) RMSE 结果对比　　(c) 仿真吞吐概率密度

图 8-10 场景 6　$CW_{min}=32$, $v=158.4$Mbps, $r=5$



(a) 吞吐量结果对比　　(b) RMSE 结果对比　　(c) 仿真吞吐概率密度

图 8-11 场景 7　$CW_{min}=16$, $v=158.4$Mbps, $r=32$

表 8-4 总结了不同竞争窗口、最大重传次数和物理层速率下仿真和模型理论解的对比，从表中可以看出，模型在 7 类场景下的平均均方根误差为 2.9633，精度较高，能够较好的反应存在部分监听和同频干扰的 3BSS 系统。

表 8-4 7 种场景下的仿真和理论解对比

	场景1	场景2	场景3	场景4	场景5	场景6	场景7
CW_{min}	16	16	32	16	16	32	16
CW_{max}	1024	1024	1024	1024	1024	1024	1024
最大重传次数r	32	6	5	32	6	5	32
物理层速率v (Mbps)	455.8	286.8	286.8	286.8	158.4	158.4	158.4
模型吞吐量 (Mbps)	108.4228	98.7974	82.8179	98.8140	82.7612	71.2892	82.7761
仿真吞吐量 (Mbps)	107.7990	99.3392	76.7326	99.2826	87.0340	67.9734	87.4568
RMSE 均方根误差	1.1726	0.2368	6.2842	0.1703	4.4003	3.9886	4.4904

九、模型评价与改进

9.1 模型的优点

本文主要研究 WLAN 网络信道接入机制，采用离散时间的二维马尔可夫模型分析 WLAN 基础结构模式中 DCF 机制的吞吐量，针对四种不同场景，考虑同频干扰、隐藏节点、节点数量和丢包等诸多因素，并运用 Matlab 软件对模型进行求解。分析节点接入的信道评估、随机回退和数据传输 3 个阶段，基于 Python 语言开发仿真器实现监听、退避、接入等功能。数值分析与仿真实验结果表明，本文所建数学模型能够恰当的模拟载波侦听多址接入/退避机制，有效的评估各种通信场景下系统的吞吐量。

9.2 模型的缺点及改进

1.本文所考虑的四个场景中，节点数量都是固定的，而在实际的通信场景中，节点规模可能随着时间而不断变化，导致更多复杂的通信问题，后续考虑分析节点数量对吞吐量的影响，并将其反映到所建数学模型中。

2.本文所考虑的四个场景中，每个节点发送的每个数据包的帧长都是固定的，而在实际的通信场景中，对于不同的任务需求，节点发送的数据包的帧长一般是不同的，后续计划将数据包帧长作为变量加入到模型中来完善模型。

3.在模型中，我们假设了接收方发送的 ACK 不会丢失，而在实际通信场景中，ACK 是可能会丢失的，后续在模型中考虑 ACK 的传输过程来拓展模型，并采用相应的方法对 ACK 的丢失进行处理。

参考文献

[1] Bianchi Giuseppe. IEEE 802.11-Saturation Throughput Analysis [J]. IEEE Communications Letters, 1998, 2(12):318-320.

[2] Bianchi Giuseppe, "Performance analysis of the IEEE 802.11 distributed coordination function," in IEEE Journal on Selected Areas in Communications, vol. 18, no. 3, pp. 535-547, March 2000, doi: 10.1109/49.840210.

[3] P. Chatzimisios, V. Vitsas and A. C. Boucouvalas, "Throughput and delay analysis of IEEE 802.11 protocol," Proceedings 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications, 2002, pp. 168-174, doi: 10.1109/IWNA.2002.1241355.

[4] Kuo W K. Energy efficiency modeling for IEEE 802.11 DCF system without retry limits [J].Com- puter Communications, 2007,30(4): 856-862.

1008−1019. http://www.springerlink.com/content/rkchd978egg2wye6/

[5] Duffy K, Malone D, Leith DJ. Modeling the 802.11 distributed coordination function in non-saturated conditions. IEEE Communications Letters, 2005,9(8):715−717.

[6] Malone D, Duffy K, Leith DJ. Modeling the 802.11 distributed coordination function in non-saturated heterogeneous conditions. IEEE/ACM Trans. on Networking, 2007,15(1):159−172.

[7] Latkoski P, Hadzi-Velkov Z, Popovski B. Extended model for performance analysis of non-saturated IEEE 802.11 DCF in erroneous channel. In: Mobile Adhoc and Sensor Systems (MASS). New York: IEEE Press, 2006. 783−788.

[8] ZHAI H, KWON YFANG Y.Performance analysis of IEEE 802.11 MAC protocols in wireless LANs[J]. Wireless Communication Mobile Computing,2004,4:917-931.

[9] Hung, Fu-Yi, and Ivan Marsic. "Performance analysis of the IEEE 802.11 DCF in the presence of the hidden stations." Computer Networks 54.15 (2010): 2674-2687.

[10] D. R. Chen and Y. J. Zhang, "Is Dynamic Backoff Effective for Multi-Rate WLANs?" in IEEE Communications Letters, vol. 11, no. 8, pp. 647-649, August 2007

[11]杨卫东,马建峰,李亚辉.基于分组到达率的 802．11DCF 性能分析[J].软件学报,2008(10):2762-2769.

[12]唐伦,刘益富,刘青海等.一种改进 IEEE 802.11 DCF 的建模与分析[J].北京邮电大学学报,2014,37(05):96-99.DOI:10.13190/j.jbupt.2014.05.020.

附录

附录 A 问题 1 求解程序

程序说明	问题 1 模型求解	编程语言	Matlab

```
CW_min=16;
CW_max=1024;
r=32;
m=log2(CW_max/CW_min);
rate=455.8;
H=8*30/rate+13.6;
SIFS=16;
ACK=32;
DIFS=43;
ACKout=65;
Ep=1500*8/rate;
xi=9;
Ts=H+Ep+SIFS+ACK+DIFS;
Tc=H+Ep+DIFS+ACKout;
syms x
sol = vpasolve(x == 2*(1-2*x)*(1-x^(r+1))/(CW_min*(1-(2*x)^(m+1))*(1-x)+(1-2*x)*(1-x^(r+1))+CW_min*2^m*x^(m+1)*(1-x^(r-m))*(1-2*x)), x);
tau=sol(1);
Ptr=1-(1-tau)^2;
Ps=2*tau*(1-tau)/Ptr;
S=Ps*Ptr*Ep/((1-Ptr)*xi+Ptr*Ps*Ts+Ptr*(1-Ps)*Tc)*rate;
disp(S)
```

程序说明	问题 1 仿真器中的 AP 类	编程语言	Python

```
import random
import AP_config as ac
class APs():
    def __init__(self, idx):
        self.idx = idx
        self.state = ac.AP_state[0]
        self.time = 0
        self.CW_count = 0
        self.CW = ac.CW[self.CW_count]
        self.fallback = 0
        self.channel = [1, 1] # 一开始信道设置为繁忙，强迫进行 CCA 阶段
        self.trans_count = 0
        self.retrans = 0
        self.is_finish = False
        self.is_success = True
    def step(self, channel):
        flag = 0
```

```
        self.time = self.time+1
        if self.time % ac.DIFS_time == 0:
            self.channel = channel
        if self.state == ac.AP_state[0]:          # CCA
            if self.channel[0] == 0 and self.channel[1] == 0:      # 信道无干扰才会进入
随机回退
                self.state = ac.AP_state[1]
                self.fallback = random.choice(range(0, ac.CW[self.CW_count])) *
ac.SLOT_time
        elif self.state == ac.AP_state[1]:        # 随机回退
            if self.fallback % ac.SLOT_time == 0:
                if self.channel[1 - self.idx] == 0 and self.fallback != 0: # 判断对方没传
                    self.fallback = self.fallback - 1
                if self.fallback == 0:
                    self.state = ac.AP_state[2]
                    self.trans_count = 0
            else:
                self.fallback = self.fallback - 1
        else:                                # 数据传输
            flag = 1
            self.trans_count = self.trans_count + 1
            if channel[1-self.idx] == 0:
                if self.trans_count > ac.H + ac.EP +ac.ACK_time+ac.SIFS_time:
                    self.is_finish = True          # 传输成功
                    flag = 0
            else:
                if self.trans_count > ac.H + ac.EP + ac.ACKTimeout:      # 数据传输失
败，等待重传
                    if self.retrans < ac.max_retrans:
                        self.state = ac.AP_state[1]           # 回到随机回退状态
                        if self.CW_count < len(ac.CW)-1:
                            self.CW_count = self.CW_count + 1
                        self.fallback = random.choice(range(1, ac.CW[self.CW_count])) *
ac.SLOT_time
                        self.retrans = self.retrans+1
                    else:
                        self.is_finish = True
                        self.is_success = False
        return flag
    def reset(self):
        self.state = ac.AP_state[0]
        self.time = 0
        self.CW_count = 0
        self.CW = ac.CW[self.CW_count]
        self.fallback = 0
        self.channel = [1, 1] # 一开始信道设置为繁忙，强迫进行 CCA 阶段
        self.trans_count = 0
        self.retrans = 0
        self.is_finish = False
```

```
        self.is_success = True
```

程序说明	问题 1 仿真器中的 config 文件	编程语言	Python

```
AP_state = ['CCA', 'randreturn', 'trans']
CW = [16, 32, 64, 128, 256, 512, 1024]
ACK_time = 32
SIFS_time = 16
DIFS_time = 43
SLOT_time = 9
ACKTimeout = 65
max_retrans = 32
H = 8*30/455.8+13.6
EP = 8*1500/455.8
```

程序说明	问题 1 仿真器中的主函数	编程语言	Python

```
from AP import APs
from tqdm import trange
import scipy.io as sio
import time
import numpy as np
import os
time_str = time.strftime("%Y%m%d_%H%M%S")
valid = 1500*8/275.3
Max_time = 100000
Max_ep = 10
mat_res_path = './Results'
alg_name = 'task_2'
rate = 275.3
Save_th = np.zeros((Max_ep, Max_time))
Save_tau = np.zeros((Max_ep, Max_time))
save_th = {
        "th": Save_th,
        "tau": Save_tau,
    }
for j in range(Max_ep):
    data = 0
    trans_num = 0
    virtual_slot = 1
    AP1 = APs(0)
    AP2 = APs(1)
    AP1_res = 0
    AP2_res = 0
    channel = [AP1_res, AP2_res] # 无 AP 传输信道空闲 0，单 AP 传输信道繁忙
1，2AP 传输失败 2
    for i in trange(Max_time):
        if not AP1.is_finish:
            AP1_res = AP1.step(channel)
        if not AP2.is_finish:
            AP2_res = AP2.step(channel)
        if AP1.is_finish:
            data = data+valid
```

```
            AP1.reset()
        if AP2.is_finish:
            data = data+valid
            AP2.reset()
        if channel != [AP1_res, AP2_res] or (channel == [0,0] and i % 9 == 0):
            virtual_slot = virtual_slot + 1
        if AP1_res == 1 and channel[0] == 0:
            trans_num = trans_num+1
        if AP2_res == 1 and channel[1] == 0:
            trans_num = trans_num+1
        channel = [AP1_res, AP2_res]
        save_th["th"][j, i] = data / (i + 1) * rate
        save_th["tau"][j, i] = trans_num/virtual_slot
        # if channel==[1,1]:
        # print(channel)
    print(data/Max_time*rate)
save_mat_path = os.path.join(mat_res_path, alg_name + "_results", time_str)
os.makedirs(save_mat_path)
mat_path = os.path.join(save_mat_path, "results.mat")
sio.savemat(mat_path,{'th': Save_th, "tau": Save_tau})
```

附录 B 问题 2 求解程序

程序说明	问题 2 模型求解	编程语言	Matlab

```
clc
clear
xi=9;
CW_min = 16;
rate=275.3;
DIFS = 43;
H = 8*30/rate+13.6;
Ep= 1500*8/rate;
SIFS=16;
ACK=32;
ACKout=65;
Ts=H+Ep+SIFS+ACK+DIFS;
syms x
sol = vpasolve(x == (x+1)/(((CW_min-1)/2)+x+1), x);
for j = 1:size(sol, 1)
    if 0<=sol(j) && 1>=sol(j) && imag(sol(j))==0
        tau=sym2poly(sol(j));
    end
end

Ptr=1-(1-tau)^2;
S=(Ptr+tau^2)*Ep/((1-Ptr)*xi+Ptr*Ts)*rate;
disp(S)
```

程序说明	问题 2 仿真器中的 AP 类	编程语言	Python

```
import random
import AP_config as ac
class APs():
    def __init__(self, idx):
        self.idx = idx
        self.state = ac.AP_state[0]
        self.time = 0
        self.CW_count = 0
        self.CW = ac.CW[self.CW_count]
        self.fallback = random.choice(range(0, 15)) * ac.SLOT_time
        self.frozen_time = ac.DIFS_time
        self.frozen_flag = False
        self.channel = [1, 1] # 一开始信道设置为繁忙，强迫进行 CCA 阶段
        self.trans_count = 0
        self.is_finish = False
    def step(self, channel):
        flag = 0
        self.time = self.time+1
        if self.time % ac.DIFS_time == 0:
            self.channel = channel
        if self.state == ac.AP_state[0]:          # CCA
            if self.channel[self.idx] == 0:   # 信道无干扰才会进入随机回退
                self.state = ac.AP_state[1]
        elif self.state == ac.AP_state[1]:        # 随机回退
            if self.fallback % ac.SLOT_time == 0:
                if channel[1-self.idx] == 1:
                    self.frozen_flag = True
                elif channel[1-self.idx] == 0 and self.frozen_flag:
                    if self.frozen_time == 0:
                        self.frozen_flag = False
                        self.frozen_time = ac.DIFS_time
                    else:
                        self.frozen_time = self.frozen_time-1
                if channel[1 - self.idx] == 0 and self.fallback != 0 and not self.frozen_flag:
# 判断对方没传
                    self.fallback = self.fallback - 1
                if self.fallback == 0:
                    self.state = ac.AP_state[2]
                    self.trans_count = 0
            else:
                self.fallback = self.fallback - 1
        else:                                # 数据传输
            flag = 1
            self.trans_count = self.trans_count + 1
            if self.trans_count >= ac.H + ac.EP + ac.ACK_time + ac.SIFS_time:
                self.is_finish = True          # 传输成功
                # flag = 0
        return flag
    def reset(self):
```

```
        self.state = ac.AP_state[0]
        self.time = 0
        self.CW_count = 0
        self.CW = ac.CW[self.CW_count]
        self.fallback = random.choice(range(0, 15)) * ac.SLOT_time
        self.channel = [1, 1] # 一开始信道设置为繁忙，强迫进行 CCA 阶段
        self.trans_count = 0
        self.frozen_time = ac.DIFS_time
        self.frozen_flag = False
        self.is_finish = False
```

程序说明	问题 2 仿真器中的 config 文件	编程语言	Python

```
AP_state = ['CCA', 'randreturn', 'trans']
CW = [16, 32, 64, 128, 256, 512, 1024]
ACK_time = 32
SIFS_time = 16
DIFS_time = 43
SLOT_time = 9
ACKTimeout = 65
rate = 275.3
H = 8*30/rate+13.6
EP = 8*1500/rate
```

程序说明	问题 2 仿真器中的主函数	编程语言	Python

```
from AP import APs
from tqdm import trange
import scipy.io as sio
import time
import numpy as np
import os
time_str = time.strftime("%Y%m%d_%H%M%S")
valid = 1500*8/275.3
Max_time = 100000
Max_ep = 10
mat_res_path = './Results'
alg_name = 'task_2'
rate = 275.3
Save_th = np.zeros((Max_ep, Max_time))
Save_tau = np.zeros((Max_ep, Max_time))
save_th = {
        "th": Save_th,
        "tau": Save_tau,
    }
for j in range(Max_ep):
    data = 0
    trans_num = 0
    virtual_slot = 1
    AP1 = APs(0)
    AP2 = APs(1)
    AP1_res = 0
    AP2_res = 0
```

```
    channel = [AP1_res, AP2_res]  # 无 AP 传输信道空闲 0，单 AP 传输信道繁忙
1，2AP 传输失败 2
    for i in trange(Max_time):
        if not AP1.is_finish:
            AP1_res = AP1.step(channel)
        if not AP2.is_finish:
            AP2_res = AP2.step(channel)
        if AP1.is_finish:
            data = data+valid
            AP1.reset()
        if AP2.is_finish:
            data = data+valid
            AP2.reset()
        if channel != [AP1_res, AP2_res] or (channel == [0,0] and i % 9 == 0):
            virtual_slot = virtual_slot + 1
        if AP1_res == 1 and channel[0] == 0:
            trans_num = trans_num+1
        if AP2_res == 1 and channel[1] == 0:
            trans_num = trans_num+1
        channel = [AP1_res, AP2_res]
        save_th["th"][j, i] = data / (i + 1) * rate
        save_th["tau"][j, i] = trans_num/virtual_slot
        # if channel==[1,1]:
        # print(channel)
    print(data/Max_time*rate)
save_mat_path = os.path.join(mat_res_path, alg_name + "_results", time_str)
os.makedirs(save_mat_path)
mat_path = os.path.join(save_mat_path, "results.mat")
sio.savemat(mat_path,{'th': Save_th, "tau": Save_tau})
```

附录 C 问题 3 求解程序

程序说明	问题 3 模型求解	编程语言	Matlab

```
clc
clear
CW_min = [16,16,32,16,16,32,16];
CW_max = 1024;
r = [32 6,5,32,6,5,32];
rate = [455.8 286.8 286.8 286.8 158.4 158.4 158.4];
V = ceil((8*(1500+30)./rate+13.6)/9);
m = log2(CW_max./CW_min);
tau1=zeros(size(CW_min, 2),1);
tau2=zeros(size(CW_min, 2),1);
xi=9;
H=8*30./rate+13.6;
SIFS=16;
ACK=32;
DIFS=43;
ACKout=65;
```

```
Ep=1500*8./rate;
Ts=H+Ep+SIFS+ACK+DIFS;
Tc=H+Ep+DIFS+ACKout;
for i=1:size(CW_min, 2)
   syms p;
   sol = vpasolve((p-0.1)/0.9 == ((V(i)+1)*(1-p^r(i)/(1-p))-(V(i)*(V(i)+1) ...
      /(2*CW_min(i)))*((1-(p/2)^(r(i)+1))/(1-(p/2))))*2*(1-p)*(1-2*p)/ ...
      ((1-2*p)*(1-p^(r(i)+1))+CW_min(i)*(1-p)*(1-(2*p)^(r(i)+1))), p);
   for j = 1:size(sol, 1)
      if 0<sol(j) && 1>sol(j) && imag(sol(j))==0
         tau2(i) = (sol(j)-0.1)/0.9;
         p1=0;
         b00=2/(CW_min(i)+1);
         tau1(i)=(1-p1^(r(i)+1))/(1-p1)*b00;
      end
   end
end
for i=1:size(CW_min, 2)
   Ptr=1-(1-tau1(i))^2;
   Ps=2*tau1(i)*(1-tau2(i))/Ptr*0.9;
   S(i)=Ps*Ptr*Ep(i)/((1-Ptr)*xi+Ptr*Ps*Ts(i)+Ptr*(1-Ps)*Tc(i))*rate(i);
end
disp(S)
```

程序说明	问题 3 仿真器中的 AP 类	编程语言	Python

```
import random
import AP_config as ac
import numpy as np
class APs():
   def __init__(self, idx, j):
      self.j = j
      self.idx = idx
      self.state = ac.AP_state[0]
      self.time = 0
      self.CW_count = 0
      self.CW = ac.CW[j][self.CW_count]
      self.fallback = 0
      self.channel = [1, 1]  # 一开始信道设置为繁忙，强迫进行 CCA 阶段
      self.trans_count = 0
      self.retrans = 0
      self.is_finish = False
      self.is_success = True
      self.is_interference = False
   def step(self, channel, channel_real):
      flag = 0                # 虚假的传输信息
      flag_real = 0           # 真实的传输信息
      self.time = self.time+1
      if self.time % ac.DIFS_time == 0:
         self.channel = channel
```

```
        if self.state == ac.AP_state[0]:            # CCA
            if self.channel[0] == 0 and self.channel[1] == 0: # 信道无干扰才会进入随机回退
                self.state = ac.AP_state[1]
                self.fallback = random.choice(range(0, ac.CW[self.j][self.CW_count])) * ac.SLOT_time
        elif self.state == ac.AP_state[1]:          # 随机回退
            if self.fallback % ac.SLOT_time == 0:
                if self.channel[1 - self.idx] == 0 and self.fallback != 0:  # 判断对方没传
                    self.fallback = self.fallback - 1
                if self.fallback == 0:
                    self.state = ac.AP_state[2]
                    self.trans_count = 0
                    self.is_interference = np.random.choice([True, False], p=[0.1, 0.9])
            else:
                self.fallback = self.fallback - 1
        else:                                       # 数据传输
            flag_real = 1                           # 数据开始传输
            self.trans_count = self.trans_count + 1
            if channel_real[1-self.idx] == 0 and not self.is_interference:
                if self.trans_count > ac.H[self.j] + ac.EP[self.j] + ac.SIFS_time:
                    flag = 1                        # ACK 广播真实传输信息
                if self.trans_count > ac.H[self.j] + ac.EP[self.j] + ac.ACK_time + ac.SIFS_time:
                    self.is_finish = True           # 传输成功
            else:
                if self.trans_count > ac.H[self.j] + ac.EP[self.j] + ac.ACKTimeout:    # 数据传输失败，等待重传
                    if self.retrans < ac.max_retrans[self.j]:
                        self.state = ac.AP_state[1]   # 回到随机回退状态
                        if self.CW_count < len(ac.CW[self.j]) - 1:
                            self.CW_count = self.CW_count + 1
                        self.fallback = random.choice(range(1, ac.CW[self.j][self.CW_count])) * ac.SLOT_time
                        self.retrans = self.retrans+1
                    else:
                        self.is_finish = True
                        self.is_success = False
        return flag, flag_real
    def reset(self):
        self.state = ac.AP_state[0]
        self.time = 0
        self.CW_count = 0
        self.CW = ac.CW[self.j][self.CW_count]
        self.fallback = 0
        self.channel = [1, 1] # 一开始信道设置为繁忙，强迫进行 CCA 阶段
        self.trans_count = 0
        self.retrans = 0
```

```
        self.is_finish = False
        self.is_success = True
        self.is_interference = False
```

程序说明	问题 3 仿真器中的 config 文件	编程语言	Python

```
import numpy as np
AP_state = ['CCA', 'randreturn', 'trans']
CW = [[16, 32, 64, 128, 256, 512, 1024],[16, 32, 64, 128, 256, 512, 1024],[32, 64, 128, 256, 512, 1024],[16, 32, 64, 128, 256, 512, 1024],
    [16, 32, 64, 128, 256, 512, 1024],[32, 64, 128, 256, 512, 1024],[16, 32, 64, 128, 256, 512, 1024]]
ACK_time = 32
SIFS_time = 16
DIFS_time = 43
SLOT_time = 9
ACKTimeout = 65
max_retrans = [32,6,5,32,6,5,32]
rate = np.array([455.8, 286.8, 286.8, 286.8, 158.4, 158.4, 158.4])
H = 8*30./rate+13.6
EP = 8*1500./rate
```

程序说明	问题 3 仿真器中的主函数	编程语言	Python

```
from AP import APs
from tqdm import trange
import scipy.io as sio
import time
import numpy as np
import os
time_str = time.strftime("%Y%m%d_%H%M%S")
rate = np.array([455.8, 286.8, 286.8, 286.8, 158.4, 158.4, 158.4])
valid = 1500*8./rate
Max_time = 100000
Max_ep = 10
mat_res_path = './Results'
alg_name = 'task_3'
Save_th = np.zeros((Max_ep, len(rate), Max_time))
Save_tau = np.zeros((Max_ep, len(rate), Max_time))
Save_p = np.zeros((Max_ep, len(rate), Max_time))
save_th = {
        "th": Save_th,
        "tau": Save_tau,
        "p": Save_p,
    }
for k in trange(Max_ep):
    for j in range(len(rate)):
        data = 0
        virtual_slot = 1
        trans_num = 1
        collision = 0
        AP1 = APs(0, j)
        AP2 = APs(1, j)
        AP1_res = 0
```

```
        AP2_res = 0
        AP1_res_real = 0
        AP2_res_real = 0
        channel = [AP1_res, AP2_res]  # 无 AP 传输信道空闲 0，单 AP 传输信道繁忙 1，2AP 传输失败 2
        channel_real = [AP1_res_real, AP2_res_real]
        for i in range(Max_time):
            if not AP1.is_finish:
                AP1_res, AP1_res_real = AP1.step(channel, channel_real)
            else:
                if AP1.is_success:
                    data = data+valid[j]
                AP1.reset()
            if not AP2.is_finish:
                AP2_res, AP2_res_real = AP2.step(channel, channel_real)
            else:
                if AP2.is_success:
                    data = data+valid[j]
                AP2.reset()
            if channel_real != [AP1_res_real, AP2_res_real] or (channel_real == [0, 0] and i % 9 == 0):
                virtual_slot = virtual_slot + 1
            if (AP1_res_real == 1 and channel_real[0] == 0) or (AP2_res_real == 1 and channel_real[1] == 0):
                trans_num = trans_num + 1
            if (AP1_res_real == 1 and AP2_res_real == 1 and channel_real[1] == 0):
                collision = collision + 1
            channel = [AP1_res, AP2_res]
            channel_real = [AP1_res_real, AP2_res_real]
            save_th["th"][k, j, i] = data / (i + 1) * rate[j]
            tau = trans_num / virtual_slot
            p = collision / trans_num
            save_th["tau"][k, j, i] = tau
            save_th["p"][k, j, i] = p
        # print(data/Max_time*rate[j])
save_mat_path = os.path.join(mat_res_path, alg_name + "_results", time_str)
os.makedirs(save_mat_path)
mat_path = os.path.join(save_mat_path, "results.mat")
sio.savemat(mat_path,{'th': Save_th, "tau": Save_tau})
```

附录 D 问题 4 求解程序

程序说明	问题 4 模型求解	编程语言	Matlab

```
clc
clear
CW_min = [16 16 32 16 16 32 16];
CW_max = 1024;
r = [32 6 5 32 6 5 32];
rate =[455.8 286.8 286.8 286.8 158.4 158.4 158.4];
```

```
m1 = log2(CW_max./CW_min);
tau1=zeros(size(CW_min, 2),1);
tau2=zeros(size(CW_min, 2),1);
S=zeros(size(CW_min, 2),0);
xx=zeros(size(CW_min,2),4);
xi=9;
H=8*30./rate+13.6;
SIFS=16;
ACK=32;
DIFS=43;
ACKout=65;
Ep=1500*8./rate;
Ts=H+Ep+SIFS+ACK+DIFS;
Tc=H+Ep+DIFS+ACKout;
for i=1:length(CW_min)
   x=fsolve(@(x)f(x,r(i),m1(i),CW_min(i)),[0.1 0.1 0.1 0.1]);
   xx(i,:)=x;
   tau1=x(1);
   tau2=x(2);
   p1=x(3);
   p2=x(4);
   Ptr=1-(1-tau2)*(1-tau1);
   Ps=(2*(1-tau2)*tau1+(1-tau1)*tau2)/Ptr;
   S(i)=(Ps)*Ptr*Ep(i)/((1-Ptr)*xi+Ptr*Ps*Ts(i)+Ptr*(1-Ps)*Tc(i))*rate(i);
end
disp(S)
function y = f(x, r,m,w)
   f1=1-(1-x(1))^2-x(4);
   f2=x(2)-x(3);
   if r<=m
      f3=          2*(1-x(3))*(1-2*x(3))/((1-2*x(3))*(1-x(3)^(r+1))+w*(1-x(3))*(1-
(2*x(3))^(r+1)))*(1-x(3)^(r+1))/(1-x(3))-x(1);
      f4=          2*(1-x(4))*(1-2*x(4))/((1-2*x(4))*(1-x(4)^(r+1))+w*(1-x(4))*(1-
(2*x(4))^(r+1)))*(1-x(4)^(r+1))/(1-x(4))-x(2);
   else
      f3=          2*(1-x(3))*(1-2*x(3))/((1-2*x(3))*(1-x(3)^(r+1))+w*(1-x(3))*(1-
(2*x(3))^(m+1))+w*2^m*x(3)^(m+1)*(1-x(3)^(r-m)*(1-2*x(3))))*(1-
x(3)^(r+1))/(1-x(3))-x(1);
      f4=          2*(1-x(4))*(1-2*x(4))/((1-2*x(4))*(1-x(4)^(r+1))+w*(1-x(4))*(1-
(2*x(4))^(m+1))+w*2^m*x(4)^(m+1)*(1-x(4)^(r-m)*(1-2*x(4))))*(1-
x(4)^(r+1))/(1-x(4))-x(2);
   end
   y=[f1 f2 f3 f4];
end
```

程序说明	问题 4 仿真器中的 AP1 和 AP3 类	编程语言	Python

```
import random
import AP_config as ac
import numpy as np
class APs_1():
```

```
    def __init__(self, idx, j):
        self.j = j
        self.idx = idx
        self.state = ac.AP_state[0]
        self.time = 0
        self.CW_count = 0
        self.CW = ac.CW[self.CW_count]
        self.fallback = 0
        self.channel = [1, 1]  # 一开始信道设置为繁忙，强迫进行 CCA 阶段
        self.trans_count = 0
        self.retrans = 0
        self.is_finish = False
        self.is_success = True
    def step(self, channel):
        flag = 0
        self.time = self.time+1
        if self.time % ac.DIFS_time == 0:
            self.channel = channel
        if self.state == ac.AP_state[0]:          # CCA
            if self.channel[1] == 0:      # 信道无干扰才会进入随机回退
                self.state = ac.AP_state[1]
                self.fallback = random.choice(range(0, ac.CW[self.j][self.CW_count])) *
ac.SLOT_time
        elif self.state == ac.AP_state[1]:        # 随机回退
            if self.fallback % ac.SLOT_time == 0:
                if self.channel[1] == 0 and self.fallback != 0: # 判断对方没传
                    self.fallback = self.fallback - 1
                if self.fallback == 0:
                    self.state = ac.AP_state[2]
                    self.trans_count = 0
            else:
                self.fallback = self.fallback - 1
        else:                                  # 数据传输
            flag = 1                          # 数据开始传输
            self.trans_count = self.trans_count + 1
            if channel[1] == 0:
                if self.trans_count > ac.H[self.j] + ac.EP[self.j] + ac.ACK_time +
ac.SIFS_time:
                    self.is_finish = True         # 传输成功
                    flag = 0
            else:
                if self.trans_count > ac.H[self.j] + ac.EP[self.j] + ac.ACKTimeout:    # 数
据传输失败，等待重传
                    if self.retrans < ac.max_retrans[self.j]:
                        self.state = ac.AP_state[1]   # 回到随机回退状态
                        if self.CW_count < len(ac.CW[self.j]) - 1:
                            self.CW_count = self.CW_count + 1
                        self.fallback                =                random.choice(range(1,
ac.CW[self.j][self.CW_count])) * ac.SLOT_time
```

```
                self.retrans = self.retrans+1
            else:
                self.is_finish = True
                self.is_success = False
        return flag
    def reset(self):
        self.state = ac.AP_state[0]
        self.time = 0
        self.CW_count = 0
        self.CW = ac.CW[self.j][self.CW_count]
        self.fallback = 0
        self.channel = [1, 1, 1] # 一开始信道设置为繁忙，强迫进行 CCA 阶段
        self.trans_count = 0
        self.retrans = 0
        self.is_finish = False
        self.is_success = True
```

程序说明	问题 4 仿真器中的 AP2 类	编程语言	Python

```
import random
import AP_config as ac
import numpy as np
class APs_2():
    def __init__(self, idx, j):
        self.j = j
        self.idx = idx
        self.state = ac.AP_state[0]
        self.time = 0
        self.CW_count = 0
        self.CW = ac.CW[self.CW_count]
        self.fallback = 0
        self.channel = [1, 1, 1] # 一开始信道设置为繁忙，强迫进行 CCA 阶段
        self.trans_count = 0
        self.retrans = 0
        self.is_finish = False
        self.is_success = True
    def step(self, channel):
        flag = 0                # 传输信息
        self.time = self.time+1
        if self.time % ac.DIFS_time == 0:
            self.channel = channel
        if self.state == ac.AP_state[0]:          # CCA
            if self.channel[0] == 0 and self.channel[2] == 0:      # 信道无干扰才会进入
随机回退
                self.state = ac.AP_state[1]
                self.fallback = random.choice(range(0, ac.CW[self.j][self.CW_count])) *
ac.SLOT_time
        elif self.state == ac.AP_state[1]:        # 随机回退
            if self.fallback % ac.SLOT_time == 0:
                if self.channel[0] == 0 and self.channel[2] == 0 and self.fallback != 0:  #
判断 2 个覆盖节点没传
```

```
                self.fallback = self.fallback - 1
            if self.fallback == 0:
                self.state = ac.AP_state[2]
                self.trans_count = 0
        else:
            self.fallback = self.fallback - 1
    else:
        flag = 1                    # 数据开始传输
        self.trans_count = self.trans_count + 1
        if channel[0] == 0 and channel[2] == 0:
            if self.trans_count > ac.H[self.j] + ac.EP[self.j] + ac.ACK_time + ac.SIFS_time:
                self.is_finish = True       # 传输成功
                flag = 0
        else:
            if self.trans_count > ac.H[self.j] + ac.EP[self.j] + ac.ACKTimeout:    # 数据传输失败，等待重传
                if self.retrans < ac.max_retrans[self.j]:
                    self.state = ac.AP_state[1]  # 回到随机回退状态
                    if self.CW_count < len(ac.CW[self.j]) - 1:
                        self.CW_count = self.CW_count + 1
                    self.fallback = random.choice(range(1, ac.CW[self.j][self.CW_count])) * ac.SLOT_time
                    self.retrans = self.retrans+1
                else:
                    self.is_finish = True
                    self.is_success = False
    return flag
def reset(self):
    self.state = ac.AP_state[0]
    self.time = 0
    self.CW_count = 0
    self.CW = ac.CW[self.CW_count]
    self.fallback = 0
    self.channel = [1, 1, 1] # 一开始信道设置为繁忙，强迫进行 CCA 阶段
    self.trans_count = 0
    self.retrans = 0
    self.is_finish = False
    self.is_success = True
```

程序说明	问题 4 仿真器中的 config 文件	编程语言	Python

```
import numpy as np
AP_state = ['CCA', 'randreturn', 'trans']
CW = [[16, 32, 64, 128, 256, 512, 1024],[16, 32, 64, 128, 256, 512, 1024],[32, 64, 128, 256, 512, 1024],[16, 32, 64, 128, 256, 512, 1024],
    [16, 32, 64, 128, 256, 512, 1024],[32, 64, 128, 256, 512, 1024],[16, 32, 64, 128, 256, 512, 1024]]
ACK_time = 32
SIFS_time = 16
DIFS_time = 43
```

```
SLOT_time = 9
ACKTimeout = 65
max_retrans = [32,6,5,32,6,5,32]
rate = np.array([455.8, 286.8, 286.8, 286.8, 158.4, 158.4, 158.4])
H = 8*30./rate+13.6
EP = 8*1500./rate
```

程序说明	问题 4 仿真器中的主函数	编程语言	Python

```
import numpy as np
from AP_1 import APs_1
from AP_2 import APs_2
from tqdm import trange
import scipy.io as sio
import time
import numpy as np
import os
rate = np.array([455.8, 286.8, 286.8, 286.8, 158.4, 158.4, 158.4])
valid = 1500*8./rate
Max_time = 1000000
Max_ep = 10
mat_res_path = './Results'
alg_name = 'task_4'
time_str = time.strftime("%Y%m%d_%H%M%S")
Save_th = np.zeros((Max_ep, len(rate), Max_time))
save_th = {
      "th": Save_th
   }
for k in trange(Max_ep):
   for j in range(len(rate)):
      data = 0
      AP1 = APs_1(0, j)     # 一个隐藏，一个覆盖
      AP2 = APs_2(1, j)     # 两个覆盖节点
      AP3 = APs_1(2, j)
      AP1_res = 0
      AP2_res = 0
      AP3_res = 0
      channel = [AP1_res, AP2_res, AP3_res]
      for i in range(Max_time):
         if not AP1.is_finish:
            AP1_res = AP1.step(channel)
         else:
            if AP1.is_success:
               data = data+valid[j]
            AP1.reset()
         if not AP2.is_finish:
            AP2_res = AP2.step(channel)
         else:
            if AP2.is_success:
               data = data+valid[j]
            AP2.reset()
```

```
            if not AP3.is_finish:
                AP3_res = AP3.step(channel)
            else:
                if AP3.is_success:
                    data = data+valid[j]
                AP3.reset()
            save_th["th"][k, j, i] = data / (i + 1) * rate[j]
            channel = [AP1_res, AP2_res, AP3_res]
        print(data/Max_time*rate[j])
save_mat_path = os.path.join(mat_res_path, alg_name + "_results", time_str)
os.makedirs(save_mat_path)
mat_path = os.path.join(save_mat_path, "results.mat")
sio.savemat(mat_path,{"th": Save_th})
```