


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	同济大学
参赛队号	**23102470073**
队员姓名	**1.**王陈瑁
	2.马新一
	3.张映泽

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）所提供的通信服务质量，可以参考**吞吐**这一指标， WLAN 较高的吞吐量可以在某种程度上反应该网络的通信较为便捷。随着移动互联网等通信技术应用场景的增多，如手机、物联设备等站点（STA）和无线路由器等无线接入点（AP）这两类节点的部署密度也日益增加，使 STA 关联于 AP 所得基本服务集（BSS）相互干扰的概率增大，导致网络吞吐量降低。

对于多节点共享信道进行同频数据传输的场景，分布式协调功能（DCF）机制提供了一种各节点避免冲突依次接入的方法，其模式有载波侦听多址接入/退避（CSMA/CA）机制。Bianchi 模型使用 Markov chain 对单 BSS 的 DCF 机制进行建模，可以求出其所有传输状态的稳态概率，从而进一步计算网络的吞吐量，该方法具有较高的精确度。本文对于 CSMA/CA 机制在不同类型的应用场景，首先使用 Bianchi 模型对吞吐量进行计算，后通过离散事件仿真方法，对该场景进行仿真模拟，从而对 Bianchi 模型的精确度进行验证。

对于**问题一**：两 BSS 互听系统存在同频干扰情形，标准 Bianchi 模型数值分析方法求解结果为$\mathbf{S = 6.72\times10^7 bps}$，改进的 Bianchi 模型结果为$\mathbf{S = 6.60\times10^7 bps}$，仿真器模拟结果均值为$\mathbf{\bar{S} = 6.35\times10^7 bps}$，改进的 Bianchi 模型结果与仿真更为接近，仅相差 **3.8%**。

对于**问题二**：两 BSS 互听系统存在终端接收数据 SIR 较高，两个 AP 数据传输都能成功的情形，此时不存在重传。构建 Markov chain 利用数值分析方法求解结果为$\mathbf{S = 7.06\times10^7 bps}$，仿真器求解结果为$\mathbf{\bar{S} = 6.72\times10^7 bps}$，相差 **4.8%**。

对于**问题三**：两 BSS 不互听系统存在 AP 发包在时间上有交叠时 SIR 较小，导致两 AP 发包均失败的情形。此问题需要分类讨论交叠时的状态，包括处于成功传输的状态和传输失败的状态。传统的 Bianchi 模型简化了这两个状态，不适用于本问题，因此在传统 Bianchi 模型中引入了成功传输状态和传输失败状态，更贴近真实情况。数值分析方法求解结果为$\mathbf{S = 5.45\times10^7 bps}$，仿真器求解结果为$\mathbf{\bar{S} = 5.55\times10^7 bps}$，仅相差 **1.8%**。

对于**问题四**：三 BSS 系统存在 AP1 与 AP3 不互听，AP2 与两者都互听，SIR 较大使得 AP1 和 AP3 发包时间交叠也都发送成功的情形。本问题结合了问题一和问题三两种情景，其中不互听的两 AP 具有对称性，可以同时建立 Markov chain 模型进行分析，而另一个 AP 的性质与另外两个 AP 不同，因此需要单独建立 Markov chain 模型进行分析。最后将两个模型的参数联立求解系统内各节点的稳态概率。数值分析方法求解结果为$\mathbf{S = 9.76\times10^7 bps}$，仿真器求解结果为$\mathbf{\bar{S} = 1.10\times10^8 bps}$，相差 **12.27%**。

关键词：　DCF；CSMA/CA；Bianchi 模型；离散事件仿真

目 录

1 问题重述

1.1 问题背景

无线局域网（WLAN, wireless local area network）技术在现代社会有着越来越广泛的应用，其具有通信范围广、速度快、成本低、组建方便等优势。WLAN 技术已经在笔记本电脑、手机、平板电脑等移动端广泛应用。目前 WLAN 技术正在推动物联网应用的发展，如智能家居、智慧医疗、工业制造等场景的实现，具有非常高的应用价值与广阔的应用前景。提高 WLAN 技术的通信速率和通信质量，是 WLAN 技术发展的重要一环[1]。

WLAN 基本组成部分是基本服务集（BSS，basic service set）。在一个特定的覆盖区域内，包括站点（STA，station）和一个专门管理 BSS 的无线接入点（AP，access point）。STA 可以是手机、笔记本电脑、物联网设备等，而 AP 通常是无线路由器、Wi-Fi 热点等设备。在这个系统中，AP 向 STA 发送数据称为下行方向，STA 向 AP 发送数据称为上行方向。本文将 AP 和 STA 统称为节点，并且每个节点的发送和接收是互斥的。多个节点共享一个信道，通过载波侦听多址接入/退避（CSMA/CA，carrier sense multiple access with collision avoidance）机制来避免冲突，这被称为分布式协调功能（DCF，distributed coordination function）。每个节点发出的信号都会形成一个通信区域，只有位于通信区域内的节点才能互相听到彼此的信号。

CSMA/CA 机制可以将节点在信道中进行接入传输数据的过程分为三个阶段，分别为信道可用评估（CCA，clear channel assessment）阶段、随机回退阶段和数据传输阶段。吞吐量可以在某种程度上反应 CSMA/CA 机制进行数据传输的效率高低，其计算方式为信道的利用率和物理层速率的乘积。随着 WLAN 的广泛应用，在许多场景中 AP 的部署密度会增加。考虑所有 AP 使用同一种信道的情况，当同一信道上的多个 AP 的通信区域发生重叠时，就会发生同频干扰问题。同频干扰是 WLAN 组网最值得注意的干扰问题之一，会导致网络的性能和吞吐量有一定程度的降低。

Bianchi 在 1998 年对于单 BSS，将 DCF 机制使用 Markov chain 进行模型建构，该模型假设信道理想，不会因信道质量对数据传输产生干扰。该模型的 Markov chain 的所有状态具有稳态解，根据该性质可以推导出信道在几种虚拟时隙的稳态概率，进一步计算 BSS 的吞吐量。Bianchi 模型对于其对应的传输场景具有很高的精确度，后续也有许多工作在该模型的基础上进行扩展，从而可以对不同的应用场景的吞吐量进行计算。

此外，对于真实场景中多 BSS 的同频数据传输，可以看作一种离散系统，总是可以找到某一时间点来标记系统的变化，信道接入状态是在一个个离散的时间点上发生变化。因此，通过寻找节点接入信道在离散时间点上的变化规律，通过计算机程序编程对该过程进行离散事件仿真（DES，Discrete Event Simulation）模拟是可行的。

1.2 问题重述

根据上文所述问题背景，题目围绕多同频 BSS 以 CSMA/CA 机制进行信道接入数据传输问题，设定是否互听，是否存在隐藏节点等不同的应用场景，对不同系统进行建模以评估其系统的吞吐。本文将解决下列四个问题：

(1) 两个 AP 分别向各自关联的 STA 下行数据，组成两个 BSS，AP 之间存在互听，并发传输时两个 STA 都接收失败，对该场景进行建模以评估系统的吞吐。

(2) 两个 AP 分别向各自关联的 STA 下行数据，组成两个 BSS，AP 之间存在互听，并

发传输时两个 STA 都接收成功，对该场景进行建模以评估系统的吞吐。

(3) 两个 AP 分别向各自关联的 STA 下行数据，组成两个 BSS，AP 之间不互听（互为隐藏节点），其 STA 在两个 AP 发包存在交叠时会接收失败，且因信道质量不理想会存在一定的丢包率，对该场景进行建模以评估系统的吞吐。

(4) 三个 AP 分别向各自关联的 STA 下行数据，组成三个 BSS，其中 AP2 会与 AP1 和 AP3 互听，但 AP1 与 AP3 不互听（互为隐藏节点），AP1 与 AP2 或 AP3 与 AP2 并发传输时对应 STA 都接收失败，AP1 和 AP3 发包存在交叠时不影响对应 STA 的接收。对该场景进行建模以评估系统的吞吐。

四个问题所对应的场景可概括为图 1.1，不同的 BSS 有各自的范围圈，AP 进入另一个圈范围内则会与该圈 AP 互听，STA 在另一圈范围内则可能受该圈 AP 信号干扰而接收失败。

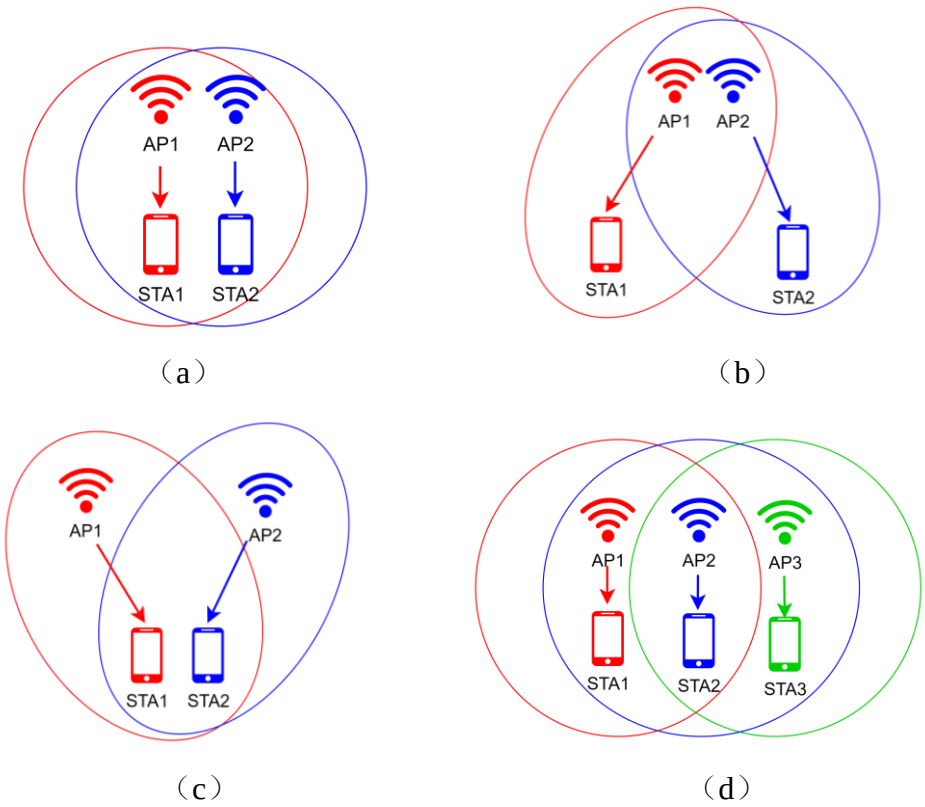


图 1.1 对应场景：（a）问题一 （b）问题二（c）问题三（d）问题四

1.3 解题思路

根据上述问题，本文解题思路如图 1.2 所示。对第一问和第二问，利用 Bianchi 模型求解各状态的稳态概率，根据系统中时间的构成计算信道吞吐量。对问题三和问题四，利用改进的 Bianchi 模型求解计算系统吞吐量。为了验证模型的有效性，四个问题均采用离散事件仿真进行验证。

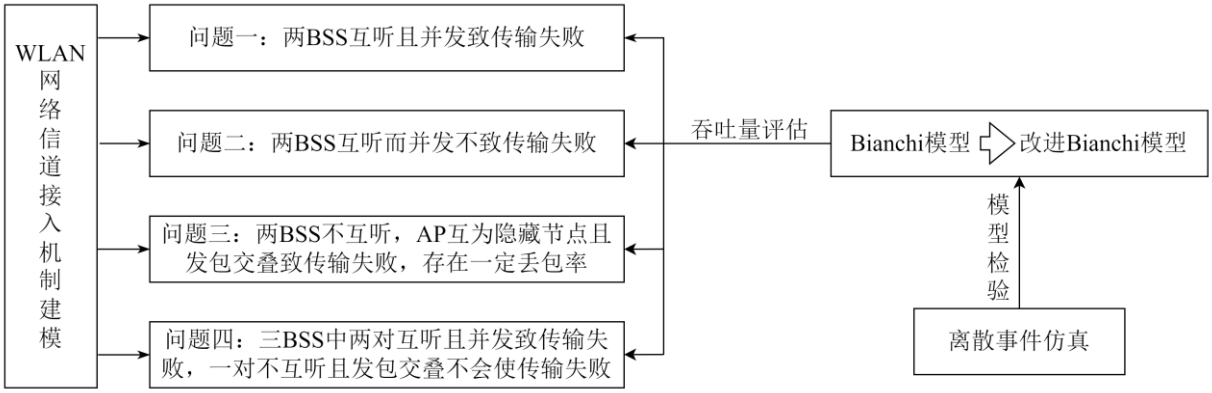


图 1.2 解题思路

2 模型假设与符号说明

2.1 模型假设

(1) 所有节点具有相同长度的数据帧。
(2) 该场景中只有问题所需要的节点，没有其他额外节点的频率干扰。
(3) 所有节点在同一频率进行数据传输。
(4) 所有节点拥有同样的发送速率。

2.2 符号说明

表 1.1 符号说明

符号	定义
p_{act}	实际传输失败条件概率
τ	节点在某时隙传输数据的概率
T_s	成功传输时长
T_e	单个回退时隙时长
$p_{overlap}$	发包交叠条件概率
T_c	传输失败时长

3 基本模型

3.1Bianchi 模型

Bianchi 在 1998 年根据二维 Markov chain 对于理想同一信道[2-5]，多 STA 给单 AP 上行数据场景所建立的模型，以阶数和退避次数这两个维度建立状态转移矩阵，该 Markov chain 具有常返性，能够求出其中所有状态的稳态解，进而可以计算数据发送成功，碰撞以及信道空闲的时间期望值，以简单易用的方法求出该场景下的信道吞吐量。基本的 Bianchi 模型如图 3.1 所示。本文从基本的 Bianchi 模型出发，通过增加状态对其进行了相应改进，后又通过仿真模拟对改进模型的效果进行测试验证。

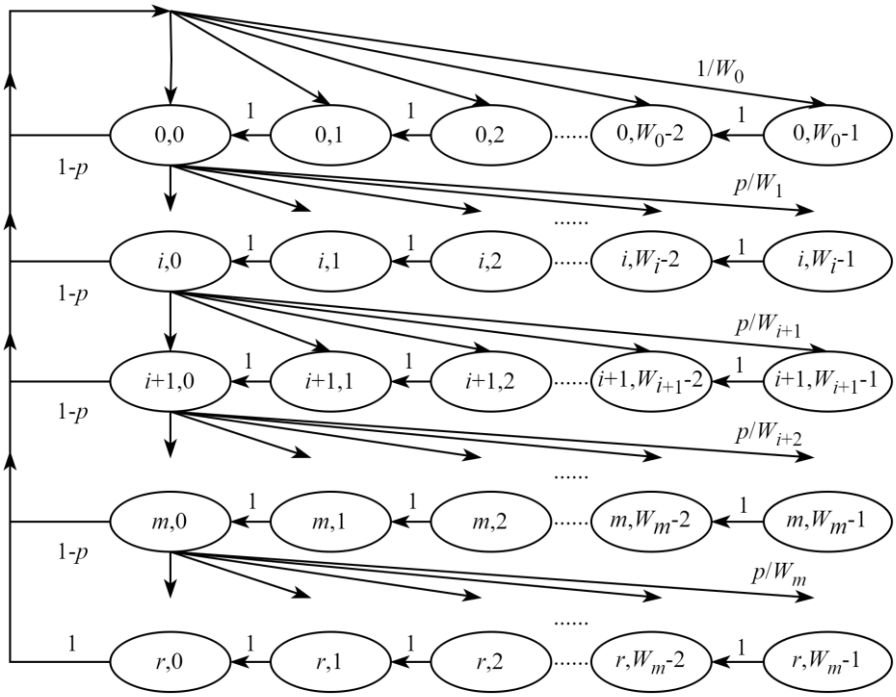


图 3.1 Bianchi 模型

3.2 离散事件仿真

离散事件仿真是一种抽象模型，将某一系统的变化看成一个事件，系统会保持原状态直到相关事件的出现，在两个事件之间，系统是一个稳定状态。

具体来说，离散与连续对应，若一个系统为离散的，则系统的变化最终能够找到一个时间点来标注，而这些标注的时间点就对应着各类事件的发生，事件出现的规律为系统变化预测的依据[6-9]。

对本文来说，在本文给定条件下的 CSMA/CA 机制信道接入问题能够抽象为一系列的离散事件，后文将使用计算机编写仿真器，将不同场景抽象为离散事件进行模拟求解。

4 问题一、二的分析与求解

4.1 问题分析

问题一为两 BSS 互听的情形，当一个 AP 传输数据时，另一 AP 通过 DIFS 侦测到 RSSI 的提高，判断信道为忙从而进入 hold time。当两 AP 同时回退到 0 时，会产生同频干扰，SIR 较低，两 AP 数据传输均失败。此问题可以通过 Bianchi 模型中 Markov chain 公式推导，求得在一个时隙发送数据的概率 τ 和并发传输的条件概率 p，最后分析系统中各部分时间的构成，计算吞吐量。

问题二为两 BSS 互听的另一种情形，与问题一区别在于当两 AP 同时回退到 0 时，并发时两个终端接收到数据的 SIR 较高，两 AP 的数据可以并发传输成功，因此没有重传。此问题需通过 Bianchi 模型推导，并分析系统中各部分时间占比并计算吞吐量。

4.2 问题一求解

4.2.1 数值求解

根据题干计算系统中的基本参数，如式（4.1）至式（4.4）所示。

$$H = MAC + PHY = 30\text{Bytes}/455.8\text{Mbps} + 13.6\mu\text{s} = 14.13\mu\text{s} \tag{4.1}$$

$$E^*[P] = E[P] = L_{payload} / rate = 1500\text{Bytes} / 455.8\text{Mbps} = 26.33\mu\text{s} \tag{4.2}$$

$$T_s = H + E[P] + SIFS + ACK + DIFS = 14.13 + 26.33 + 16 + 32 + 43 = 131.46\mu\text{s} \tag{4.3}$$

$$T_c = H + E^*[P] + DIFS + ACKTimeout = 14.13 + 26.33 + 43 + 65 = 148.46\mu\text{s} \tag{4.4}$$

由 Bianchi 模型中 Markov chain 推导可得两 AP 在某时隙发送数据概率 τ 如式（4.5）所示，两 AP 冲突条件概率 p 如式（4.6）所示。

$$P(AP_1) = P(AP_2) = \tau = b_{0,0} \times \frac{1 - p^{r+1}}{1 - p} \tag{4.5}$$

$$P(AP_2 \mid AP_1) = p = 1 - (1 - \tau)^{N-1} \tag{4.6}$$

Markov chain 中各状态稳态概率和为 1，即式（4.7）。

$$1 = \sum_{i=0}^{r} \sum_{k=0}^{W_i - 1} b_{i,k} = \sum_{i=0}^{r} \frac{W_i - k}{W_i} \cdot p^i \cdot b_{0,0} \tag{4.7}$$

联立式（4.5）、式（4.6）和式（4.7），当 $N = 2$，$m = 6$，$r = 32$，$W_0 = 16$ 时可得求得节点某时隙发送数据概率 $\tau = 10.46\%$，冲突条件概率 $p = 10.46\%$。

对两 BSS 系统的时间占比进行分析，时间占比概率图如图 4.1 所示。系统时间包括成功传输时间、传输失败时间和回退空闲时间。图中空白部分为回退空闲时间，淡阴影部分为成功传输时间，深色阴影为并发传输失败的时间。

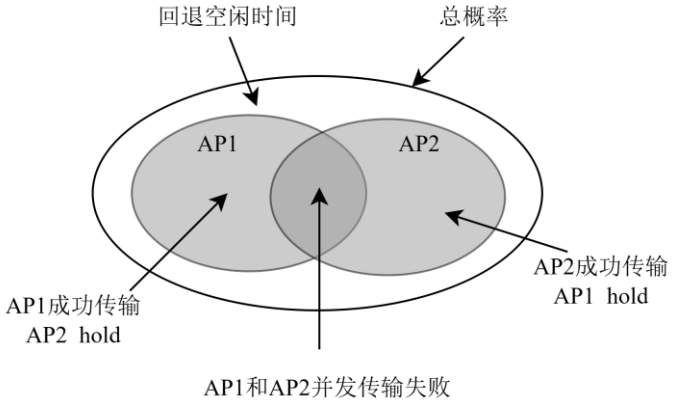


图 4.1 问题一时间分析

根据图 4.1 可计算信道吞吐量 S 如式（4.8）所示。

$$S=\frac{[P(\overline{AP_1}AP_2)+P(\overline{AP_2}AP_1)]\cdot E[P]}{P(\overline{AP_2}AP_1)T_s+P(\overline{AP_1}AP_2)T_s+P(AP_1AP_2)T_c+(1-P(AP_1)-P(AP_2)+P(AP_1AP_2))T_e}\cdot rate$$
$$=\frac{2\tau(1-p)E(P)}{2\tau(1-p)T_s+p\tau T_c+(1-2\tau+\tau p)T_e}\cdot rate \quad (4.8)$$

根据式（4.8）代入参数计算得吞吐量 $S=6.72\times10^7$ bps 。

为更准确地描述系统所有可能的状态，对 Bianchi 模型进行改进，加入成功传输状态 Trans 和失败传输状态 Failed，如图 4.2 所示。

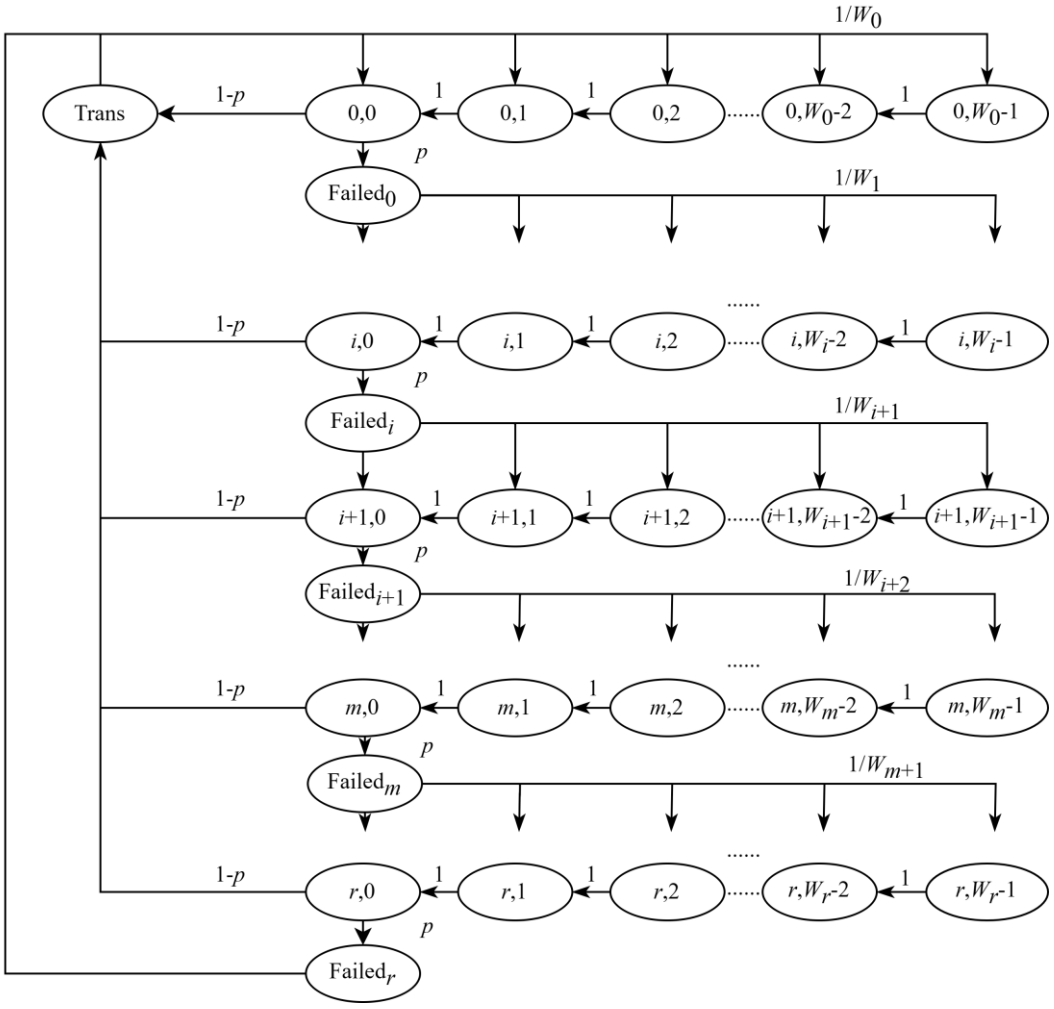


图 4.2 改进的 Bianchi 模型

改进的 Bianchi 模型状态转移概率如式（4.9）至式（4.13）所示。

$$P\{i,k\,|\,i,k+1\}=1,k\in[0,W_i-2],i\in[0,r] \quad (4.9)$$

$$P\{i,k\,|\,Failed_{i-1}\}=1/W_i,k\in[0,W_i-1],i\in[0,r] \quad (4.10)$$

$$P\{0,k\,|\,Trans\}=1/W_0,k\in[0,W_0-1],i\in[0,r] \quad (4.11)$$

$$P\{Trans\,|\,i,0\}=(1-p),i\in[0,r] \quad (4.12)$$

$$P\{Failed_i \mid i,0\} = p, i \in [0,r] \tag{4.13}$$

Trans 状态和 Failed 状态稳态概率之和如式（4.14）所示。

$$\sum_{i=0}^{r} p(Failed_i) + p(Trans) = p\sum_{i=0}^{r} b_{i,0} + (1-p)\sum_{i=0}^{r} b_{i,0} = \sum_{i=0}^{r} b_{i,0} \tag{4.14}$$

Markov chain 中状态稳态概率之和为 1，即：

$$\begin{aligned} 1 &= \sum_{i=0}^{r}\sum_{k=0}^{W_i-1} b_{i,k} + \sum_{i=0}^{r} p(Failed_i) + p(Trans) \\ &= \sum_{i=0}^{r}\sum_{k=0}^{W_i-1} \frac{W_i - k}{W_i} b_{i,0} + \sum_{i=0}^{r} b_{i,0} \\ &= \sum_{i=0}^{r} b_{i,0} \cdot \frac{W_i + 3}{2} \end{aligned} \tag{4.15}$$

联立式（4.5）、式（4.6）和式（4.15）可求得节点在某时隙发送数据概率 $\tau = 9.57\%$，冲突条件概率 $p = 9.57\%$。代入式（4.8）可求得吞吐量 $S = 6.60 \times 10^7$ bps。

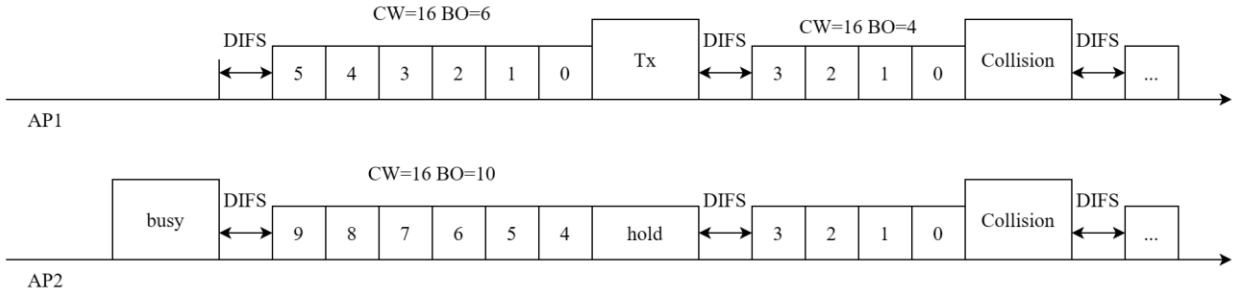
4.2.2 仿真检验

离散事件仿真由状态驱动，需要分析不同场景下状态的变化。对于问题一，以图 4.3 为例，说明该场景下 CSMA/CA 机制的运行规律。

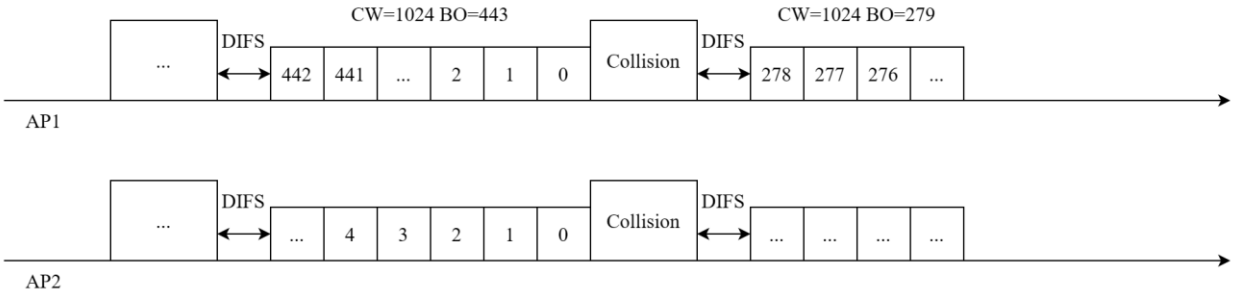
(a) 当某一 AP 数据发送次数不大于最大退避阶数时：若两个 AP 不同时回退到 0，先回退至 0 的 AP 由对应的 STA 成功接收数据，并重置其数据发送次数；后回退至 0 的 AP 则暂停回退，等待下一轮数据发送。若同时回退到 0，则发生冲突，该 AP 对应的 STA 接收失败，该 AP 数据发送次数加 1，退避窗口翻倍。

(b) 当某一 AP 数据发送次数大于最大退避阶数并小于最大重传次数时：若两个 AP 不同时回退到 0，规律与情形（a）相同。若同时回退到 0，则发生冲突，该 AP 对应的 STA 接收失败，该 AP 数据发送次数加 1，但退避窗口不翻倍。

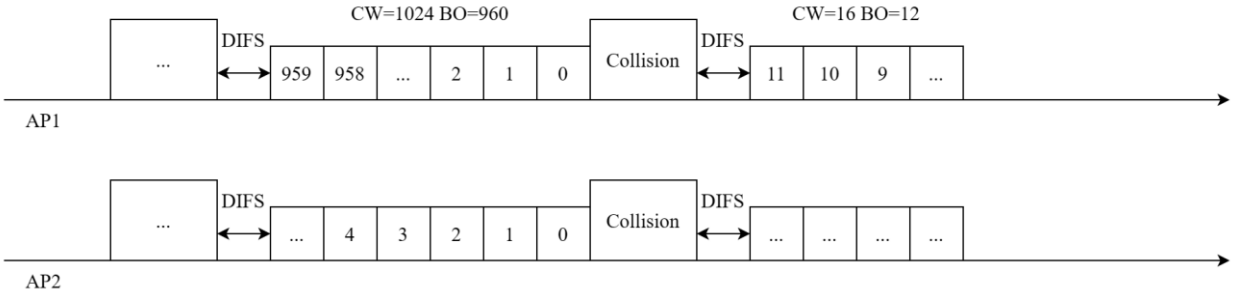
(c) 当数据发送次数等于最大重传次数时：若两 AP 不同时回退到 0，规律与情形（a）相同。若同时回退到 0，则发生冲突，该 AP 对应的 STA 接收失败，数据发送次数清 0，退避窗口重置为最小值。



（a）



（b）



（c）

图 4.3 问题一的三种情形：（a）$i \le m$ （b）$m < i < r$ （c）$i = r$

根据该场景的 CSMA/CA 机制的数据传输规律，可以将其进行离散事件仿真建模，伪代码如算法 1 所示。第 1 行至 4 行为参数初始化，其中仿真总时长 t 进入循环前已经经过一个 DIFS 时长。第 5 行开始进入离散事件循环，第 6 至 11 行为 AP 数据传输次数对退避阶数的影响，通过判断数据发送次数对退避翻倍阶数进行更新，第 12 至 16 行对应两 AP 不同时回退至 0 的情况。第 17 至 21 行对应两 AP 同时回退至 0 的情况。第 23 行在多轮仿真循环后进行吞吐量计算。

算法 1：问题一离散事件仿真伪代码	
输入:	最大退避阶数 m；最大重传次数 r；单个空闲时隙长度 T_e；成功传输时长 T_s；碰撞传输时长 T_c；DCF 帧间距 $DIFS$；仿真循环轮数 P；有效载荷传输时长 EP；物理层速率 $rate$
中间变量:	离散事件仿真总时长 t；离散事件仿真阶段数 $p \in P$；AP_a 数据的发送次数 i_a（$a \in \{1,2\}$）；AP_a 退避翻倍阶数 j_a（$a \in \{1,2\}$）；AP_a 回退次数 CW_a（$a \in \{1,2\}$）；两 AP 总体冲突次数 c
输出:	信道吞吐量 S
01	**Begin**:
02	$t = DIFS$
03	$i_a = 0,\ a \in \{1,2\} \forall a \in \{1,2\}$
04	$c = 0$
05	**For** p = 1 To P :
06	**If** $i_a \le m : j_a = i_a, \forall a \in \{1,2\}$

07 **Elif** $m < i_a < r : j_a = m, \forall a \in \{1,2\}$ //超过最大退避阶数
08 **Elif** $i_a = r :$ //达到最大重传次数
09 $j_a = 0, \forall a \in \{1,2\}$
10 $i_a = 0, \forall a \in \{1,2\}$
11 **Else**: Pass

12 **If** $CW_a < CW_b :, \forall (a,b) \in \{(1,2),(2,1)\}$ // AP_a 先回退至 **0**，成功传输
13 $t = t + (CW_a + 1) \times T_e + T_s$
14 $CW_a = random[0, 2^{j_a} \times CW_{\min} - 1]$ //更新 CW_a
15 $CW_b = CW_b - CW_a - 1$ // AP_b 剩余回退次数
16 $i_a = 0$ //重置 AP_a 数据发送次数
17 **Else**: //两 AP 同时回退至 **0**，冲突致传输失败
18 $t = t + (CW_a + 1) \times T_e + T_c$
19 $CW_a = random[0, 2^{j_a} \times CW_{\min} - 1], \forall a \in \{1,2\}$ //更新 CW_a
20 $i_a = i_a + 1, \forall a \in \{1,2\}$
21 $c = c + 1$
22 **End For**

23 **Return** $S = (P - c) \times EP \times rate / t$ //输出该场景吞吐量
24 **End Begin**

为了验证模型精确度，本文基于 Python3.9 编程语言编写仿真器实现，使用 Minitab 软件进行数值实验，每组参数下执行 40 次实验，研究在不同参数取值条件下，实验结果是否符合正态分布（显著性水平 $\alpha = 0.05$），并计算仿真器与 Bianchi 模型实验误差。其中，为检验仿真结果的稳定性，选择 Anderson–Darling Test（简称为 A-D 检验）进行正态性检验，该检验主要通过计算数据的累积分布曲线与理想正态分布的累积分布曲线之间的差异来进行检验，与 K-S 检验不同，该方法考虑了两条累积分布曲线之间的所有差异，因此它比 K-S 检验效果更好。问题一和问题二的实验结果和分析在正文中呈现，问题三和问题四中题干给出参数实验结果和分析在正文中呈现，其余在附录中呈现。

问题一实验结果如表 4.1 所示，统计计算结果如图 4.4 所示，均值为 $\overline{S} = 6.35 \times 10^7$ bps，标准差为 $\sigma = 3.3 \times 10^4$。均值 $\overline{S}$ 与改进的 Bianchi 模型求解结果相差 3.8%，与标准 Bianchi 模型求解结果相差 5.2%，证明了 Bianchi 模型的准确性和改进的有效性。

令零假设 H_0：服从正态分布。备择假设 H_1：不服从正态分布。如果 P 值小于 0.05，就拒绝 H_0，如果 P 值大于 0.05，则无法拒绝原假设 H_0。概率图如图 4.5 所示，使用 Minitab 计算出 P 值为 0.908 大于 0.05，实验结果服从正态分布，仿真结果具有稳定性。

表 4.1 问题一仿真器实验结果

单位：bps

序号	实验结果	序号	实验结果	序号	实验结果	序号	实验结果
1	63444845	11	63458518	21	63410382	31	63426032
2	63510458	12	63461696	22	63457135	32	63482346
3	63484186	13	63461720	23	63480368	33	63424432

续表 4.1

序号	实验结果	序号	实验结果	序号	实验结果	序号	实验结果
4	63465513	14	63486721	24	63536453	34	63460503
5	63503238	15	63459019	25	63440390	35	63466257
6	63522304	16	63473151	26	63511917	36	63487918
7	63481612	17	63511838	27	63427197	37	63433300
8	63481312	18	63447060	28	63475107	38	63474218
9	63492921	19	63480052	29	63419072	39	63467952
10	63448867	20	63502604	30	63500712	40	63503953

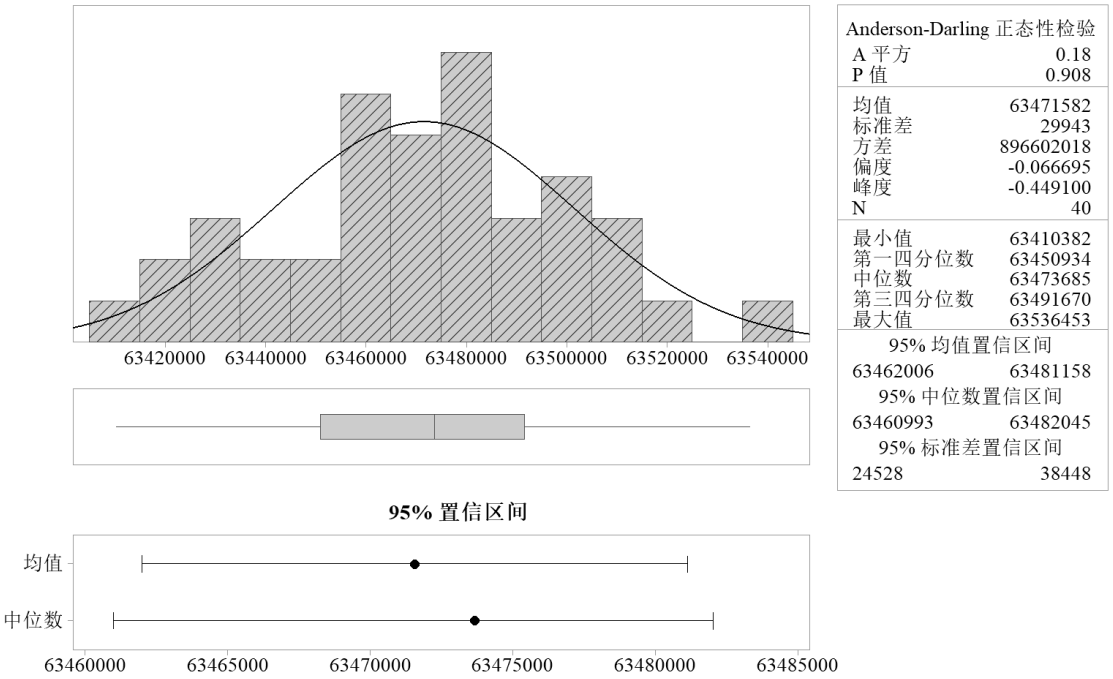


图 4.4 问题一仿真结果正态性检验报告

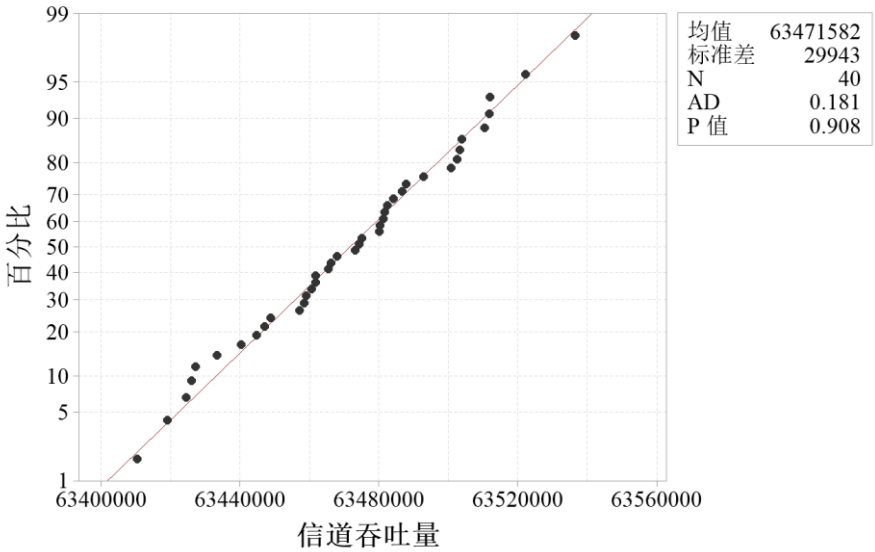


图 4.5 问题一仿真结果概率图

4.3 问题二求解

4.3.1 数值求解

根据题干计算系统的基本参数，如式（4.16）至式（4.19）所示。

$$H = MAC + PHY = 30\text{Bytes}/275.3\text{Mbps} + 13.6\mu\text{s} = 14.47\mu\text{s} \tag{4.16}$$

$$E^*[P] = E[P] = L_{payload} / rate = 1500\text{Bytes} / 455.8\text{Mbps} = 43.59\mu\text{s} \tag{4.17}$$

$$T_s = H + E[P] + SIFS + ACK + DIFS = 58.06 + 16 + 32 + 43 = 149.06\mu\text{s} \tag{4.18}$$

$$T_c = H + E^*[P] + DIFS + ACKTimeout = 14.13 + 26.33 + 43 + 65 = 166.06\mu\text{s} \tag{4.19}$$

问题二为互听的两 BSS 系统的另一种场景，由于并发时 SIR 较大，传输总是成功，因此不存在重传，Markov chain 如图 4.6 所示。

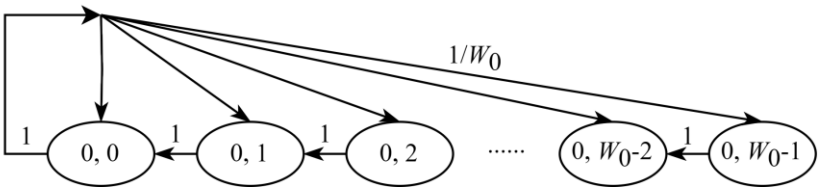


图 4.6 问题二 Markov chain

该 Markov chain 状态转移概率如式（4.20）和式（4.21）所示。

$$P\{0,k|0,0\} = 1/W_0, k \in [0, W_0 - 1] \tag{4.20}$$

$$P\{0,k|0,k+1\} = 1, k \in [0, W_0 - 1] \tag{4.21}$$

Markov chain 中各状态之和为 1，如式（4.22）所示。

$$1 = \sum_{k=0}^{W_0-1} b_{0,k} = \sum_{k=0}^{W_0-1} b_{0,0} \frac{W_0 - k}{W_0} = b_{0,0} \frac{W_0 + 1}{2}, 0 \le k \le W_0 - 1 \tag{4.22}$$

由式（4.22）可得 AP 在任意时隙发送数据的概率 τ 如式（4.23）所示。

$$P(AP_1) = P(AP_2) = \tau = b_{0,0} = 2/(W_0 + 1) \tag{4.23}$$

并发传输条件概率 p 如式（4.24）所示。

$$P(AP_2 | AP_1) = p = 1 - (1 - \tau)^{N-1} \tag{4.24}$$

当 $N = 2$，$W_0 = 16$ 时，解得 $\tau = p = 2/17$。

对当前系统的时间组成进行分析，如图 4.7 所示。图中阴影部分均为成功传输数据的时间。

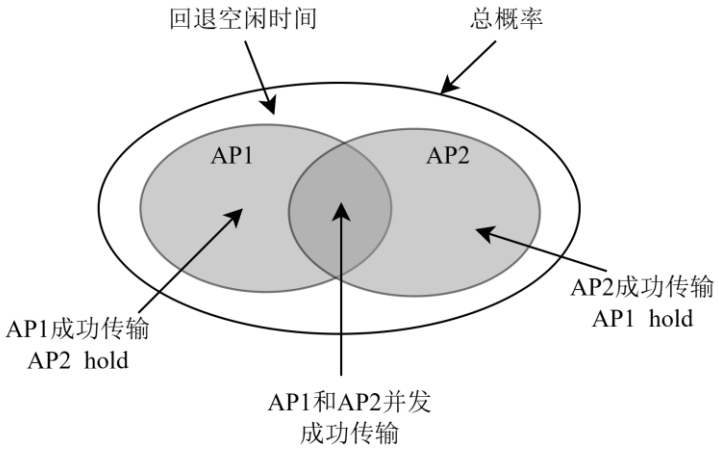


图 4.7 问题二时间分析

由图 4.7 可计算信道吞吐量 S 如式（4.25）所示。

$$\begin{aligned} S &= \frac{[P(AP_1)+P(AP_2)]\cdot E[P]}{[P(\overline{AP_2}AP_1)+P(\overline{AP_1}AP_2)+P(AP_1AP_2)]T_s+(1-P(AP_1)-P(AP_2)+P(AP_1AP_2))T_e}\cdot rate \\ &= \frac{2\tau E[P]}{2\tau(1-p)T_s+\tau pT_s+(1-2\tau+\tau p)T_e}\cdot rate \end{aligned} \tag{4.25}$$

根据式（4.25）代入参数可计算得 $S=7.06\times10^7$ dps 。

4.3.2 仿真检验

对于问题二场景下 CSMA/CA 机制的运行规律如图 4.8 所示，在该场景下，若两 AP 不同时回退至 0，则先至 0 的 AP 发送数据，另一 AP 暂停回退，等待下一轮数据发送。若两个 AP 同时回退至 0，虽然并发但 STA 都能够接收成功。所以该过程中两 AP 的回退窗口一直保持最小值，数据发送次数也在每轮置 0。

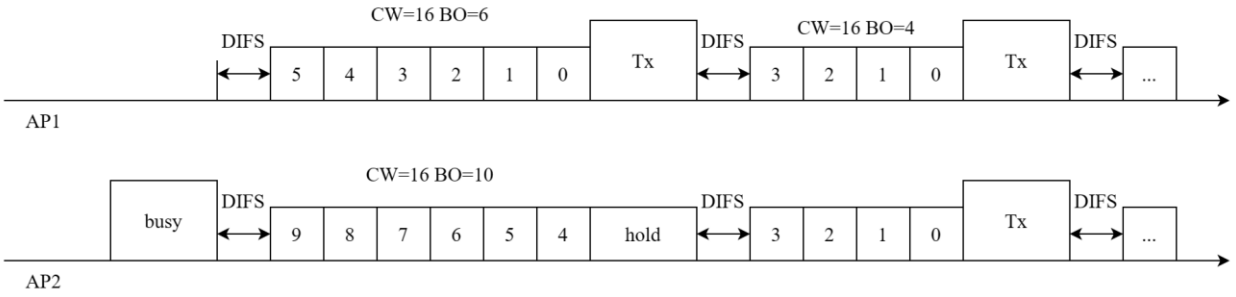


图 4.8 问题二的情形

根据该场景的 CSMA/CA 机制的数据传输规律，可以将其进行离散事件仿真建模，伪代码如算法 2 所示。第 1 至 3 行为参数初始化，第 4 行开始进入事件仿真循环，第 5 至 9 行为两 AP 不同时回退到 0 的情形，在该情形下回退到 0 的 AP 成功传输次数增加一次，第 10 至第 13 行为两 AP 同时回退至 0 并传输成功的情况，此时两 AP 成功传输次数均增加一次，第 15 行根据相关参数计算信道吞吐量。

算法 2：问题二离散事件仿真伪代码	
输入:	单个空闲时隙长度T_e；成功传输时长T_s；DCF 帧间距 *DIFS*；仿真循环轮数 P；有效载荷传输时长 *EP*； 物理层速率 *rate*；
中间变量:	离散事件仿真总时长 t； 离散事件仿真阶段数 $p \in P$； AP_a 回退次数 CW_a（$a \in \{1,2\}$）； AP_a 成功传输次数 c_a（$a \in \{1,2\}$）
输出:	吞吐量 S

```
Begin:
    t = DIFS
    c_a = 0, ∀a ∈ {1,2}

    For p = 1 To P :
        If CW_a < CW_b :, ∀(a,b) ∈ {(1,2),(2,1)}    // AP_a 先回退至 0
            t = t + (CW_a + 1) × T_e + T_s
            CW_a = random[0, 2^0 × CW_min − 1]    //更新 CW_a
            CW_b = CW_b − CW_a − 1    // AP_b 剩余回退次数
            c_a = c_a + 1
        Else:    //两 AP 同时回退至 0，但不冲突
            t = t + (CW_a + 1) × T_e + T_c
            CW_a = random[0, 2^0 × CW_min − 1], ∀a ∈ {1,2}    //更新 CW_a
            c_a = c_a + 1
    End For
    Return  S = (c_a + c_b) × EP × rate / t
End Begin
```

问题二实验结果如表 4.2 所示，统计计算结果如图 4.9 所示，均值为$\overline{S} = 6.72 \times 10^7$ bps，标准差为$\sigma = 3.8 \times 10^4$。均值$\overline{S}$与 Bianchi 模型求解结果相差 4.8%，证明了 Bianchi 模型的准确性。仿真结果概率图如图 4.10 所示，使用 Minitab 计算出 P 值为 0.567 大于 0.05，实验结果服从正态分布，仿真结果具有稳定性。

表 4.2 问题二仿真器实验结果

单位：bps

序号	实验结果	序号	实验结果	序号	实验结果	序号	实验结果
1	67234409	11	67194772	21	67209577	31	67221801
2	67182053	12	67231289	22	67227662	32	67258448
3	67187967	13	67223897	23	67244503	33	67271876
4	67286002	14	67254716	24	67232115	34	67240098
5	67259242	15	67253674	25	67211106	35	67159159
6	67219516	16	67196454	26	67158727	36	67219064
7	67222347	17	67164451	27	67111946	37	67259271
8	67199432	18	67218354	28	67260686	38	67206443
9	67247405	19	67272018	29	67274035	39	67237442
10	67193636	20	67213725	30	67172417	40	67164319

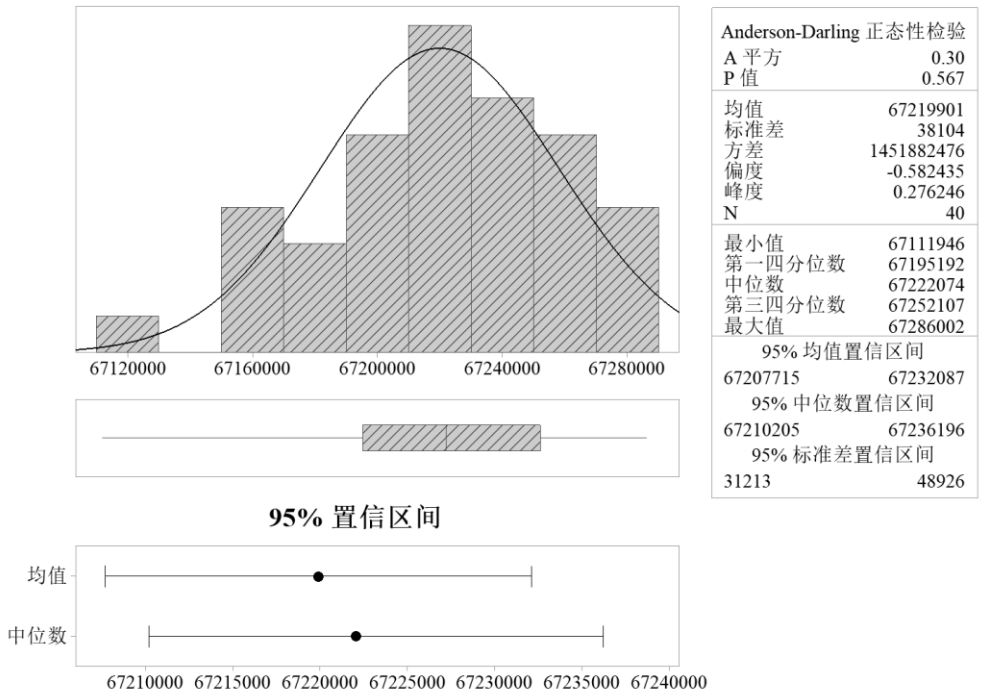


图 4.9 问题二仿真结果正态性检验报告

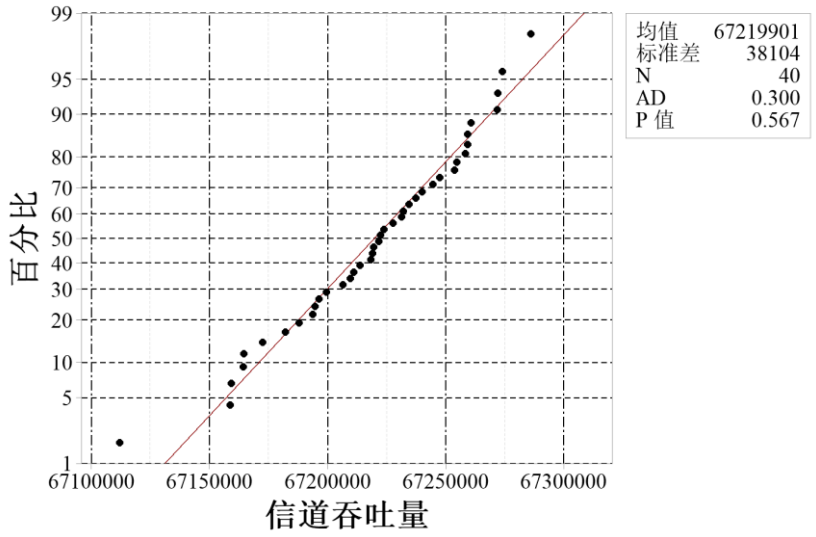


图 4.10 问题二仿真结果概率图

5 问题三的分析与求解

5.1 问题分析

问题三考虑隐藏节点问题，由于两 AP 间 RSSI 为–90dBm 低于 CCA 门限–82dBm，不互听。当 AP 发包时间 $H+E[P]$ 存在交叠时，SIR 较小，发包均失败。问题三与问题一、二的区别在于两 AP 不互听，此时 DIFS 判断信道空闲，不存在 hold time，当一个 AP 数据

传输时另一 AP 可以回退，使发包时间并非总是同时开始，因此会产生时间交叠。同时本问题还考虑了丢包的概率。

发包时间的交叠关系需要在连续时间轴上分析交叠发生时的各种情况。在本问题中连续的时间轴被分为 3 种状态，即回退时间T_e，成功传输时间T_s和失败传输时间T_c，当分析当一个 AP 在 t 时刻发包时，另一 AP 在 t 时刻处于何种状态会产生交叠时，需要对这 3 种状态中各种可能的情况进行分类讨论。由于传统 Bianchi 模型的 Markov chain 仅保留了回退时间T_c的状态，对成功传输时间T_s和失败传输时间T_c进行了简化，无法得到这两种状态的稳态概率，因此需要对 Bianchi 模型进行改进，加入成功传输状态和失败传输状态。在 Markov chain 求解获得各状态稳态概率，并根据交叠时的状态分析求解交叠条件概率 p 和时隙传输数据帧概率τ后，需要对该系统的时间组成进行分析，并最终得到吞吐量模型。

5.2 问题求解

5.2.1 数值求解

问题三考虑隐藏节点发包交叠问题，由于信号包的交叠发生在信号传输过程中，因此需重点研究两节点信号在传输状态的先后关系，但传统的 Bianchi 模型中简化了节点传输信号的状态，不适用于本问题。因此本问题的求解在传统 Bianchi 模型的基础上，在 Markov chain 中加入了信号包成功传输状态 Trans 和信号包失败传输状态 Failed，改进的 Bianchi 模型如图 5.1 所示。

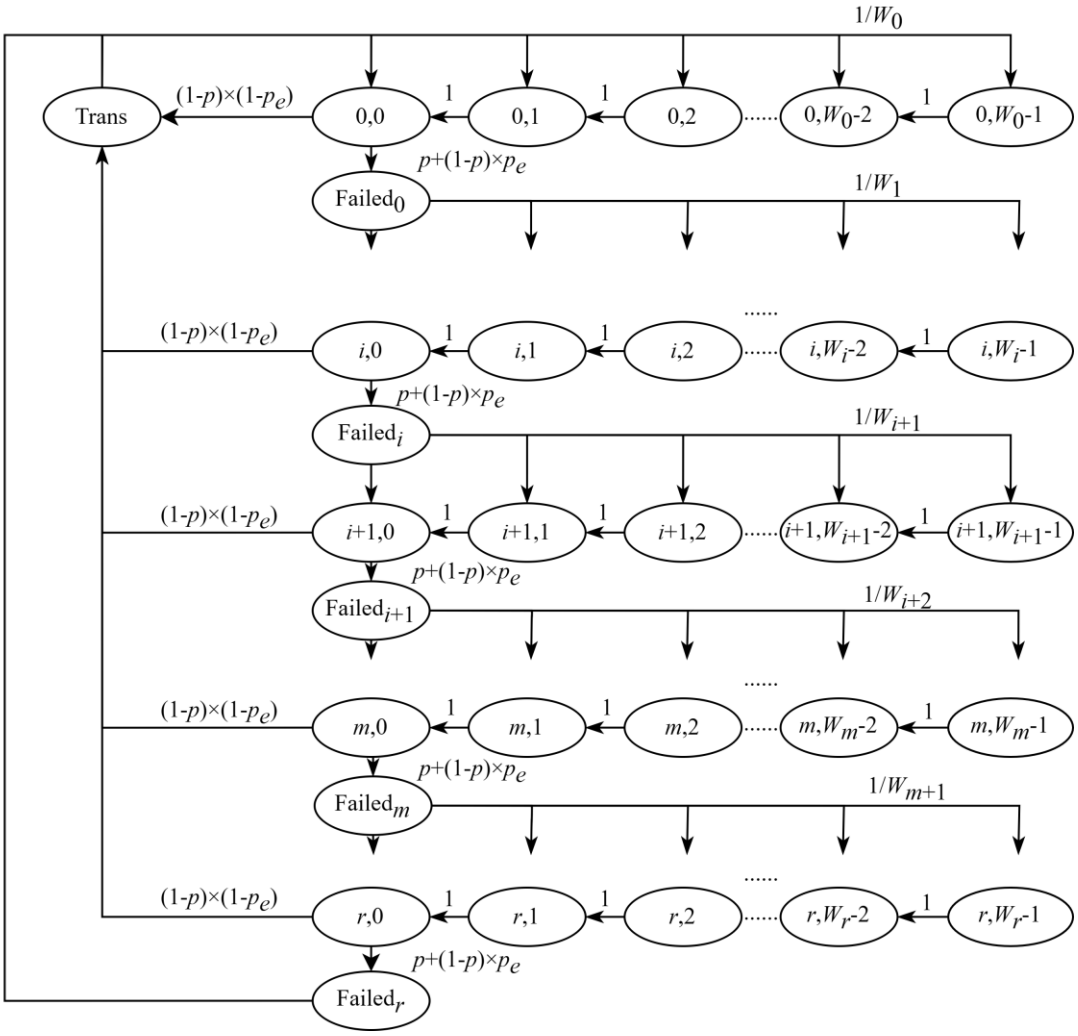


图 5.1　改进的 Bianchi 模型

由题定义可知：

$$W_i = \begin{cases} 2^i W_0, & 0 \le i \le m \\ 2^m W_0, & m \le i \le r \end{cases} \tag{5.1}$$

图 5.1 所示 Markov chain 状态转移概率如式（5.2）至式（5.6）所示

$$P\{i,k \,|\, i,k+1\} = 1, k \in [0, W_i - 2], i \in [0,r] \tag{5.2}$$

$$P\{i,k \,|\, Failed_{i-1}\} = 1/W_i, k \in [0, W_i - 1], i \in [0,r] \tag{5.3}$$

$$P\{0,k \,|\, Trans\} = 1/W_0, k \in [0, W_0 - 1], i \in [0,r] \tag{5.4}$$

$$P\{Trans \,|\, i,0\} = (1-p)(1-p_e), i \in [0,r] \tag{5.5}$$

$$P\{Failed_i \,|\, i,0\} = p + (1-p)p_e, i \in [0,r] \tag{5.6}$$

令 $b_{i,k} = \lim_{t\to\infty} P\{s(t) = i, b(t) = k\}, i \in (0,m), k \in (0, W_i - 1)$ 为各状态稳态概率， $P(Trans)$ 为传

输成功的稳态概率，$P\{Failed\}$为传输失败的稳态概率，为方便求解，令p_{act}表示考虑交叠和丢包后的实际传输失败率，如式（5.7）所示：

$$p_{act} = p + (1-p)p_e \tag{5.7}$$

根据 Markov chain 状态转移概率，可得：

$$b_{i,0} = b_{i-1,0}[W_i \cdot \frac{p_{act}}{W_i}] = b_{i-1,0}p_{act} = p_{act}^i b_{0,0}, i \in [0,r] \tag{5.8}$$

$$b_{i,k} = \begin{cases} b_{i-1,0} \cdot p_{act} \cdot \dfrac{W_i - k}{W_i}, & 0 < i < r \\ \dfrac{W_i - k}{W_i} \cdot \left[(1-p_{act}) \cdot \sum_{j=0}^{r-1} b_{j,0} + b_{r,0} \right], & i = 0 \end{cases} \tag{5.9}$$

由式（5.8）和式（5.9）可得$b_{i,k}$与$b_{i,0}$的传递关系，如式（5.10）所示。

$$b_{i,k} = \frac{W_i - k}{W_i} b_{i,0}, i \in [0,r], k \in [0, W_i - 1] \tag{5.10}$$

计算传输成功和传输失败状态出现的概率之和如式（5.11）所示。

$$\sum_{i=0}^{r} p(Failed_i) + p(Trans) = p_{act} \sum_{i=0}^{r} b_{i,0} + (1-p_{act}) \sum_{i=0}^{r} b_{i,0} = \sum_{i=0}^{r} b_{i,0} \tag{5.11}$$

由 Markov chain 的性质，所有状态的稳态概率和为 1，即：

$$\begin{aligned} 1 &= \sum_{i=0}^{r} \sum_{k=0}^{W_i-1} b_{i,k} + \sum_{i=0}^{r} p(Failed_i) + p(Trans) \\ &= \sum_{i=0}^{r} \sum_{k=0}^{W_i-1} \frac{W_i - k}{W_i} b_{i,0} + \sum_{i=0}^{r} b_{i,0} \\ &= \sum_{i=0}^{r} b_{i,0} \cdot \frac{W_i + 3}{2} \end{aligned} \tag{5.12}$$

节点在一个时隙发送数据帧的概率为：

$$\tau = \sum_{i=0}^{r} b_{i,0} = b_{0,0} \cdot \frac{1 - p_{act}^{r+1}}{1 - p_{act}} \tag{5.13}$$

求解发送失败的条件概率$P(AP_2 | AP_1)$，可以首先计算两包不交叠的条件概率$P(\overline{AP_2} | AP_1)$，以下分类讨论两包不交叠的各种状态：

状态 1：当 AP1 发包时，AP2 正处于传输状态且传输成功。状态 1 中各情况的交叠关系示意图如图 5.2 所示。

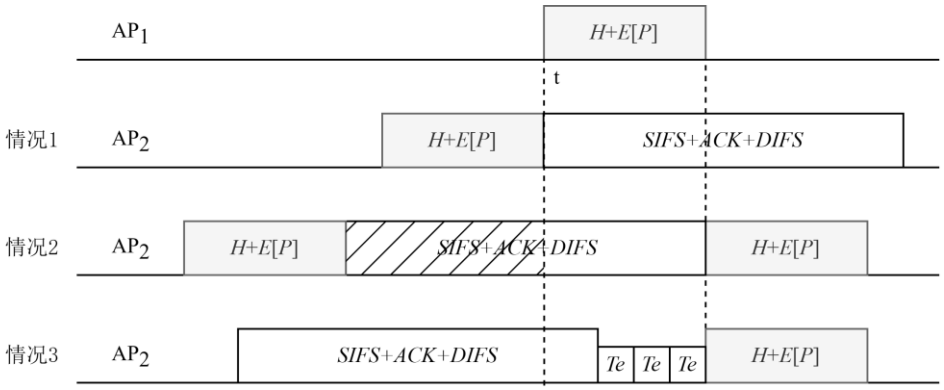


图 5.2　AP1 发包时 AP2 处于成功传输状态各情形示意图

AP1 在t时刻开始信号传输，情况 1 表示此时 AP2 恰好完成信号的有效载荷发送进入 SIFS 阶段，此时为 AP2 与 AP1 不发生交叠的极限情况。情况 2 为当 AP2 完成整个信号传输后随机到 0 个 TimeSlot，立即开始下一次信号传输，且 AP2 下一次信号传输开始的时刻恰好为 AP1 信号有效载荷传输完成的时刻，为不发生交叠的另一种极限情况。考虑两极限情况可知当 AP1 在t时刻开始信号传输时，AP2 在t时刻的状态位于数据成功传输时长T_s的前$H+E[P]$和后$H+E[P]$之间时，即图 5.2 中斜阴影部分，传输一定不会交叠。在本问题中$T_s=131.46\mu s$，$H+E[P]=40.46\mu s$，当 AP1 传输开始时，AP2 正处于T_s的时间区间$[40.46\mu s,91\mu s]$时，不会发生交叠。记该情形概率为p_1，p_1定义如式（5.14）。

$$\begin{aligned} p_1 &= p\{Trans\}\cdot\frac{T_s-2(H+E(p))}{T_s}=(1-p_{act})\sum_{i=0}^{r}b_{i,0}\cdot\frac{T_s-2(H+E(p))}{T_s} \\ &=(1-p_{act})\cdot\sum_{i=0}^{r}p_{act}^{i}b_{0,0}\cdot\frac{T_s-2(H+E(p))}{T_s} \end{aligned} \tag{5.14}$$

对于t时刻 AP2 正处于数据成功传输时长T_s的后$H+E[P]$的情况，是否交叠取决于 AP2 下一次发包随机得到 TimeSlot 的数量。例如情况 3 中，当第二次发包需要 3 个 TimeSlot 时不会交叠，若将其减少到 2 个，则会发生交叠。记$H+E[P]$中可容纳 TimeSlot 的个数为n（n为连续实数），则：

$$n=\frac{H+E[P]}{T_e} \tag{5.15}$$

信息成功传输后，随机得到 TimeSlot 个数的概率均为$1/W_0$，因此当t处于数据成功传输时长T_s的后$H+E[P]$时，AP2 在 t 时刻所指的位置越靠近T_s末尾，则越可能发生交叠，因为在下一次传输前需要更多的 TimeSlot 填补 AP1 剩余有效载荷的传输时间，因此在后$H+E[P]$段不发生交叠的概率密度函数线性下降，由 1 下降至$(W_0-n)/W_0$，如图 5.3 所示。

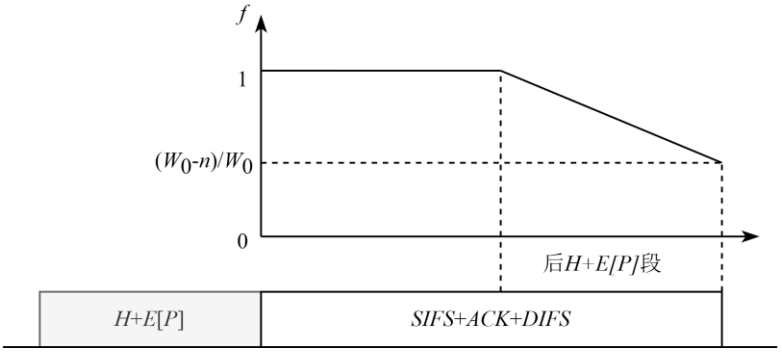


图 5.3　AP2 在 t 时刻处于成功传输状态时各处的概率密度函数

设上述情况下不发生交叠的条件概率为 p_2，p_2 定义如式（5.16）。

$$p_2 = p\{Trans\} \cdot \frac{H+E[P]}{T_s} \cdot \frac{2W_0 - n}{2W_0} = (1-p_{act}) \sum_{i=0}^{r} p_{act}^i b_{0,0} \cdot \frac{H+E[P]}{T_s} \cdot \frac{2W_0 - n}{2W_0} \quad (5.16)$$

状态 2：当 AP1 发包时，AP2 正处于传输状态且传输失败。状态 2 与状态 1 类似，区别在于此时 AP2 传输失败，消耗时长为T_c，如图 5.4 所示。

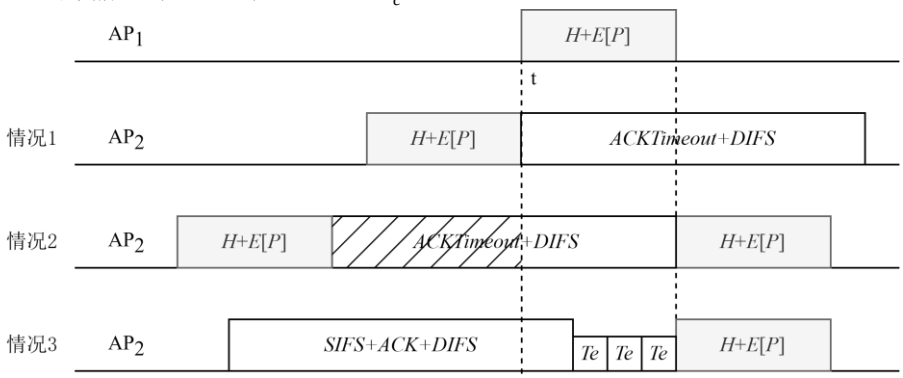


图 5.4　AP1 发包时 AP2 处于失败传输状态各情形示意图

当 AP2 在t时刻的状态位于数据失败传输时长T_c的前$H+E[P]$和后$H+E[P]$之间时，传输一定不会交叠，设此时概率为 p_3，p_3定义如式（5.17）所示。

$$\begin{aligned} p_3 &= p\{Failed\} \cdot \frac{T_c - 2(H+E(p))}{T_c} = p_{act} \sum_{i=0}^{r} b_{i,0} \cdot \frac{T_c - 2(H+E(p))}{T_c} \\ &= p_{act} \cdot \sum_{i=0}^{r} p_{act}^i b_{0,0} \cdot \frac{T_c - 2(H+E(p))}{T_c} \end{aligned} \quad (5.17)$$

对于t时刻 AP2 正处于数据失败传输时长T_c的后$H+E[P]$的情况，不发生交叠的条件概率密度函数为线性递减，设此时的概率为 p_4，p_4 定义如式（5.18）所示。

$$p_4 = p\{Failed\} \cdot \frac{H+E[P]}{T_c} \cdot \frac{2W_i - n}{2W_i} = p_{act} \sum_{i=0}^{r} p_{act}^i b_{0,0} \cdot \frac{H+E[P]}{T_c} \cdot \frac{2W_i - n}{2W_i} \quad (5.18)$$

状态 3：当 AP1 发包时，AP2 正处于 TimeSlot 回退状态。状态 3 各情况如图 5.5 所示。

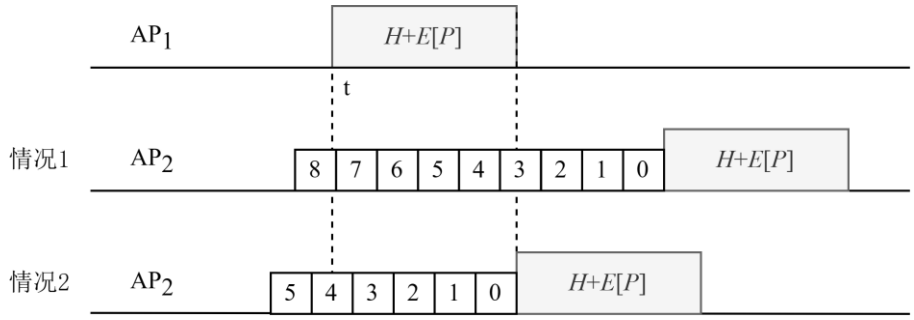


图 5.5　AP1 发包时 AP2 处于 TimeSlot 状态各情形示意图

当 AP1 在t发包时，若 AP2 处于 TimeSlot 状态，由于 TimeSlot 逐个回退，因此当t时刻 AP2 的 TimeSlot 序号大于$H+E[P]$可容纳的 TimeSlot 数n时，一定不会发生交叠。如图 5.5 情况 1 所示。若 AP1 完成有效载荷传输后，AP2 恰好开始传输，此时 AP2 在t时刻所处的 TimeSlot 为极限情况，如情况 2 所示。记此状态的不交叠条件概率为p_5，则有：

$$p_5=\sum_{i=0}^{r}\sum_{k=0}^{W_i-1}\gamma_k\cdot b_{i,k}=\sum_{i=0}^{r}\sum_{k=0}^{W_i-1}\gamma_k\cdot\frac{W_i-k}{W_i}\cdot b_{i,0}=\sum_{i=0}^{r}\sum_{k=0}^{W_i-1}\gamma_k\cdot\frac{W_i-k}{W_i}\cdot p_{act}^{i}\cdot b_{0,0} \tag{5.19}$$

式中γ_k为t时刻 AP2 指向$b_{i,k}$时不会发生交叠的概率，γ_k定义如式（5.20）所示，式中$\{\}$表示取小数部分，$\lfloor\ \rfloor$表示向下取整。

$$\gamma_k=\begin{cases}1 & ,n<k\le W_i-1\\ 1-\{n\} & ,k=\lfloor n\rfloor\\ 0 & ,0\le k<n\end{cases} \tag{5.20}$$

当$n<k\le W_i$时，表示t时刻剩余的 Timeslot 足够填满 AP1 中$H+E[P]$所需的时间，一定不会交叠，故$\gamma_k=1$。当$0<k<n$时，剩余 TimeSlot 不足以填满 AP1 中$H+E[P]$所需的时间，一定交叠，故$\gamma_k=0$，当$k=\lfloor n\rfloor$时，有$\gamma_k=1-\{n\}$的概率不会交叠，如图 5.5 中情况 2 编号为 4 的 Timeslot 所示。

综上所述，可得两 AP 发包交叠的条件概率p为：

$$P(AP_1|AP_2)=P(AP_2|AP_1)=p=1-p_1-p_2-p_3-p_4-p_5 \tag{5.21}$$

联立式（5.7）、式（5.12）、式（5.13）和式（5.21）可求得实际传输失败概率p_{act}、一个节点在某时隙发送数据帧的概率τ和两节点发包交叠条件概率p。

考虑隐藏节点问题时，两 AP 之间不存在 hold time，当 AP2 传输时，AP1 可进行指数回退，此时两 AP 系统中时间概率占比示意图如图 5.6 所示。图分为上下两层，上层为 AP1 视角的时间轴，下层为 AP2 视角的时间轴，两层时间在同一垂直方向上的点为某一时刻 AP1 和 AP2 所处的状态。

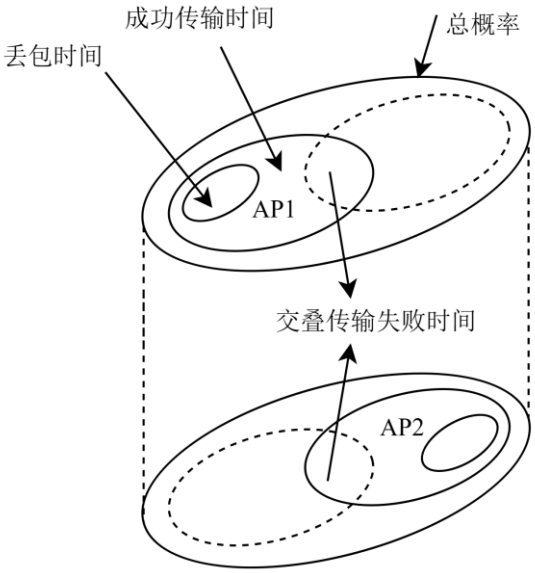


图 5.6　问题三各部分时间占比分析

根据图 5.6，在 AP1 所在时间轴上计算系统总时间，可得信道吞吐量 S 定义如式（5.22）所示。

$$\begin{aligned} S &= \frac{[(P(AP_1)-P(AP_1\overline{AP_2})-P(AP_1)p_e)+(P(AP_2)-P(AP_2\overline{AP_1})-P(AP_2)p_e)]\cdot E[p]}{(1-P(AP_1))T_e+P(AP_1)(1-P(AP_2|AP_1)-p_e)T_s+P(AP_1)(p_e+P(AP_1|AP_2))T_c}\cdot rate \\ &= \frac{2\tau(1-p-p_e+pp_e)E[P]}{(1-\tau)T_e+\tau(1-p-p_e+pp_e)T_s+\tau(p_e+p-pp_e)T_c}\cdot rate \end{aligned} \tag{5.22}$$

将参数代入式（5.22），在此条件下，利用 Markov chain 求得发包交叠条件概率 $p=26.19\%$，任意时隙发送数据帧概率 $\tau=5.91\%$，信道吞吐量 $S=5.45\times10^7$ bps。

5.2.2 仿真检验

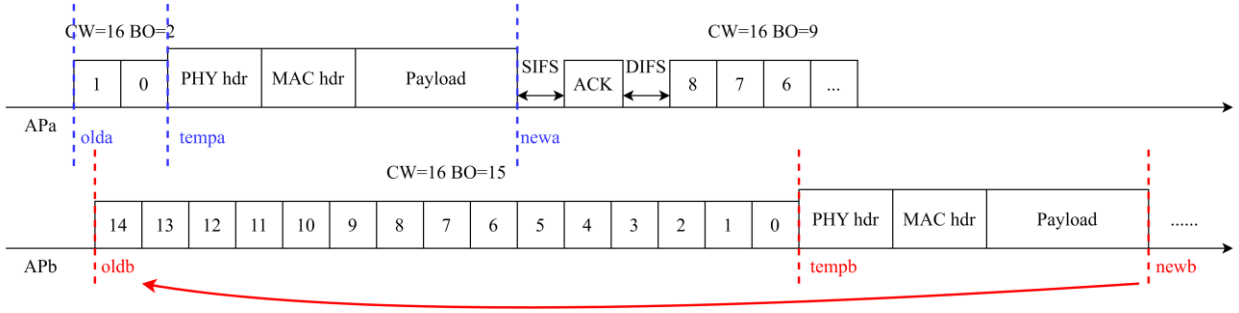
对于问题三，根据 AP_a 和 AP_b，$\forall(a,b)\in\{(1,2),(2,1)\}$ 是否同时回退至 0 将离散事件仿真模型划分为两种可能出现的情形，再根据 AP_a 和 AP_b 回退次数差异大小将情形 1 划分为情形 1-1 和情形 1-2。如图 5.7 所示，

在情形 1 中，令事件阶段为 p，AP_a 优先回退至 0，由于两 BSS 不互听，AP_b 继续回退。情形 1 可细分为情形 1-1 和情形 1-2 两种。

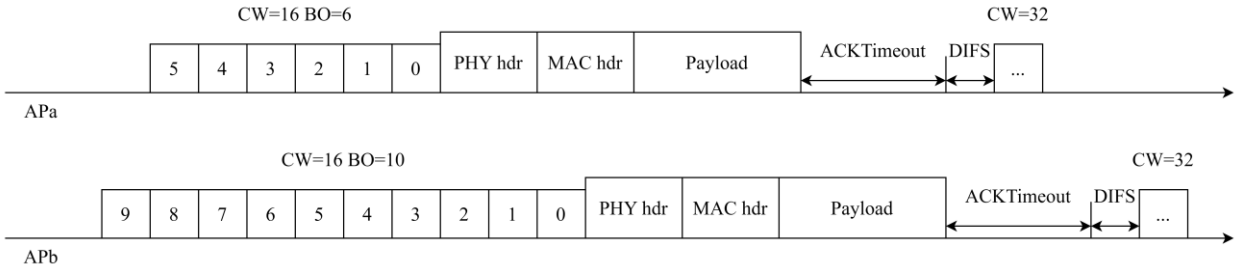
(a) 在情形 1-1 中，由于 AP_a 传输完数据后 AP_b 仍在回退过程中，AP_a 和 AP_b 不发生交叠，因此 AP_a 成功传输，AP_b 可能与下一事件阶段 $p+1$ 的 AP_a 发生交叠冲突，也可能成功传输，但均属于下一阶段需要考虑的问题，所以令 $newt_b=oldt_b$，即重置 AP_b 开始时间点，将下一事件阶段的 AP_a 与其进行比较，并重复此过程，直至 AP_b 进入下一事件阶段。

(b) 在情形 1-2 中，AP_a 数据未传输完成，AP_b 回退至 0，两者均同属一个事件阶段 p，由于 SIR 比较小，AP_a 和 AP_b 发生交叠冲突，导致两者均传输失败。

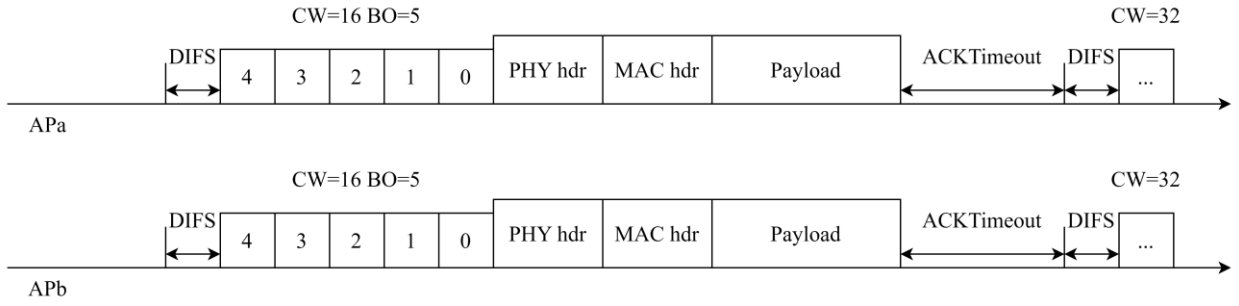
(c) 在情形 2 中，AP_a 和 AP_b 同时回退至 0，两者均传输失败。



（a）情形 1-1



（b）情形 1-2



（c）情形 2

图 5.7 问题三情形示意图（a）情形 1-1 （b）情形 1-2 （c）情形 2

问题三伪代码如算法 3 所示。第 5 行开始进入离散事件循环，其第 14 行至第 22 行对应上文情形 1-1，第 23 行至第 26 行对应情形 1-2，第 27 行至第 30 行对应情形 2，第 31 行对信道吞吐量计算结果进行输出。

算法 3：问题三离散事件仿真伪代码	
输入:	最大退避阶数 m；最大重传次数 r；单个空闲时隙长度 T_e；（PHY hdr + MAC hdr + Payload）传输时长 PMP；DCF 帧间距 $DIFS$；短帧间距 $SIFS$；仿真循环轮数 P；确认帧间距 ACK；等待超时时间 $ACKTimeout$；物理层速率 $rate$
中间变量:	回退开始时间点 $oldt_a$（$a\in\{1,2\}$）；回退结束时间点 $tempt_a$（$a\in\{1,2\}$）；传输成功或冲突失败后时间点 $newt_a$（$a\in\{1,2\}$）；离散事件仿真阶段数 $p\in P$；AP_a 数据的发送次数 i_a（$a\in\{1,2\}$）；AP_a 退避翻倍阶数 j_a（$a\in\{1,2\}$）；AP_a 回退次数 CW_a（$a\in\{1,2\}$）；AP_a 成功传输次数 s_a（$a\in\{1,2\}$）
输出:	信道吞吐量 S
01	**Begin**:
02	$oldt_a = DIFS, \forall a\in\{1,2\}$

03 $i_a = 0,\ \forall a \in \{1,2\}$

04 $s_a = 0,\ \forall a \in \{1,2\}$

05 **For** p = 1 To P :

06 **If** $i_a \le m : j_a = i_a, \forall a \in \{1,2\}$

07 **Elif** $m < i_a < r : j_a = m, \forall a \in \{1,2\}$ //超过最大退避阶数

08 **Elif** $i_a = r$: //达到最大重传次数

09 $j_a = 0, \forall a \in \{1,2\}$

10 $i_a = 0, \forall a \in \{1,2\}$

11 **Else**: Pass

12 $tempt_a = oldt_a + CW_a \times T_e, \forall a \in \{1,2\}$

13 **If** $tempt_a < tempt_b :, \forall (a,b) \in \{(1,2),(2,1)\}$ // AP_a 先回退至 **0**

14 **If** $newt_a < tempt_b$: // AP_b 回退次数过多，在当前事件阶段不发生交叠

15 $newt_b = oldt_b$ //重置为下一个事件回退开始时间点

16 **If** 丢包: $newt_a = tempt_a + PMP + ACKTimeout + DIFS$ //丢包率 **10%**

17 **Else**:

18 $newt_a = tempt_a + PMP + SIFS + ACK + DIFS$

19 $s_a = s_a + 1$

20 $CW_a = random[0, 2^{j_a} \times CW_{\min} - 1]$ // AP_a 成功传输，更新 CW_a

21 $CW_b = CW_b - CW_a - 1$ // AP_b 剩余回退次数

22 $i_a = 0$ //重置 AP_a 数据发送次数

23 **Else**: //两 AP 交叠冲突

24 $newt_a = tempt_a + PMP + ACKTimeout + DIFS$， $\forall a \in \{1,2\}$

25 $CW_a = random[0, 2^{j_a} \times CW_{\min} - 1], \forall a \in \{1,2\}$ //更新两 AP 的 CW_a

26 $i_a = i_a + 1, \forall a \in \{1,2\}$ //更新两 AP 数据发送次数

27 **Else**: //两 AP 同时回退至 **0**，冲突

28 $newt_a = tempt_a + PMP + ACKTimeout + DIFS, \forall a \in \{1,2\}$

29 $CW_a = random[0, 2^{j_a} \times CW_{\min} - 1], \forall a \in \{1,2\}$ //更新两 AP 的 CW_a

30 $i_a = i_a + 1, \forall a \in \{1,2\}$ //更新两 AP 数据发送次数

31 **Return** $S = 2 \times s_a \times EP \times rate / newt_a$ //输出该场景信道吞吐量

32 **End For**

33 **End Begin**

问题三实验结果如表 5.1 所示，统计计算结果如图 5.8 所示，均值为 $\overline{S} = 5.55 \times 10^7$ dps，标准差为 $\sigma = 9.9 \times 10^4$。与改进的 Bianchi 模型求解结果相差 1.8%，证明了改进的 Bianchi 模型的准确性。仿真结果概率图如图 5.9 所示，使用 Minitab 计算出 P 值为 0.108 大于 0.05，实验结果服从正态分布，仿真结果具有稳定性。

表 5.1 问题三仿真器实验结果

单位：dps

序号	实验结果	序号	实验结果	序号	实验结果	序号	实验结果
1	55480590	11	55413434	21	55653072	31	55527505
2	55409212	12	55517155	22	55744441	32	55695339
3	55513396	13	55546498	23	55577655	33	55619939
4	55582593	14	55658627	24	55455481	34	55469776
5	55404268	15	55751205	25	55489330	35	55442354
6	55679467	16	55431322	26	55544545	36	55710438
7	55474384	17	55528244	27	55421708	37	55595591
8	55478699	18	55391090	28	55624511	38	55512388
9	55547182	19	55490358	29	55525543	39	55559523
10	55441709	20	55467536	30	55423052	40	55628940

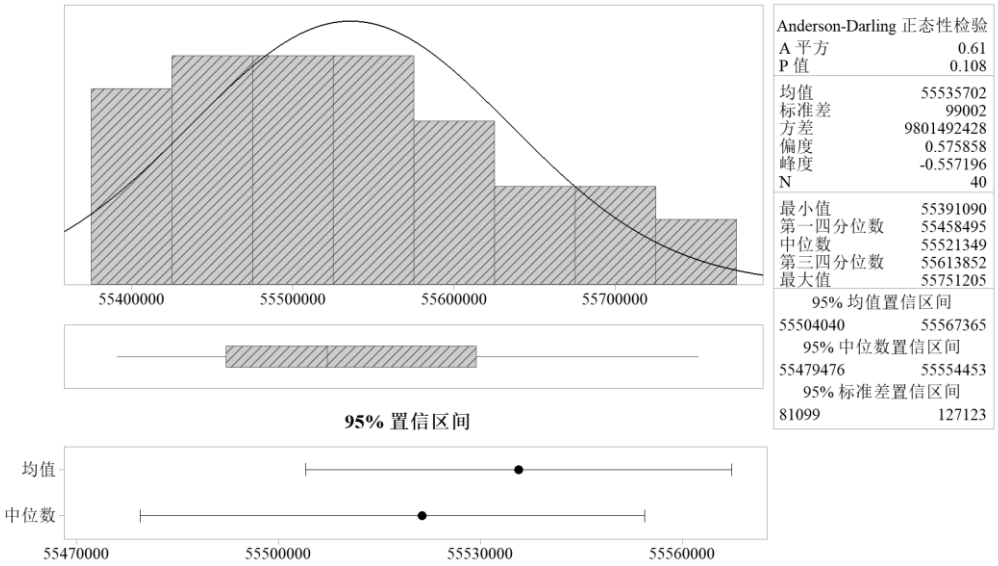


图 5.8 问题三仿真结果正态性检验报告

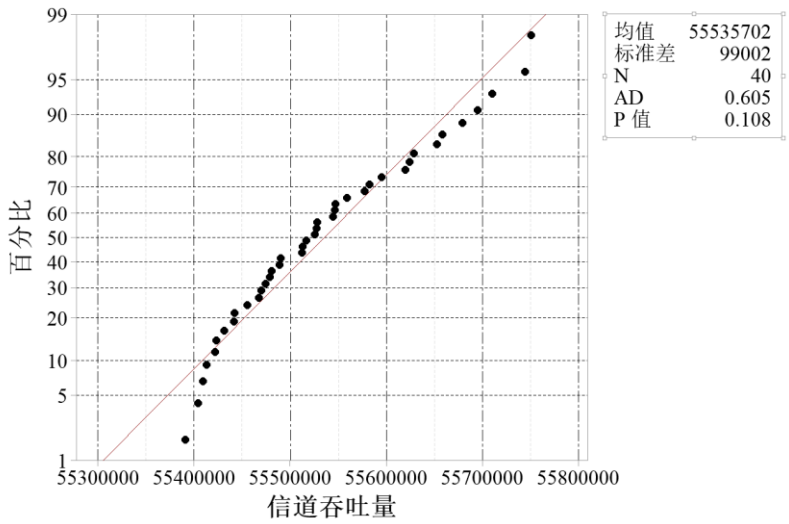


图 5.9 问题三仿真结果概率图

当系统取其他参数时，Bianchi 模型和仿真实验计算结果如表 5.2 所示。当物理层速率和最大重传次数较大时，两种方法误差在 3%以内，当物理层速率增加且最大重传次数降低时，两方法的误差略有增加，但均在 30%以内。系统取其他参数时的仿真结果正态性检验概率图见附录 A。

表 5.2 其他参数下两方法结果对比

吞吐量单位：bps

序号	CW_{min}	CW_{max}	r	物理层速率	数值计算	仿真实验	Gap
1	16	1024	32	455.8Mbps	44760000	43723195	2.31%
2	16	1024	6	286.8Mbps	41012000	32318324	21.19%
3	32	1024	5	286.8Mbps	44762000	43741428	2.28%
4	16	1024	32	286.8Mbps	32515000	28382805	12.7%
5	16	1024	6	158.4Mbps	30787000	21562020	29.96%
6	32	1024	5	158.4Mbps	32678000	28361848	13.20%

6 问题四的分析与求解

6.1 问题分析

本问题为问题一和问题三的结合，AP2 与另外两个 AP 互听，AP1 和 AP3 不互听，AP2 的传输机会可能被 AP1 和 AP3 抢占，而 AP1 和 AP3 会先后或同时发送数据，造成数据交叠，但交叠时信息能成功传输。类似于问题三，当考虑交叠问题时需要在 Bianchi 模型中加入 Trans 状态和 Failed 状态。

问题一、二和三所讨论的两 BSS 问题中，两 AP 具有对称性，但在本问题中，由于 AP2 与另外两个 AP 的状态不同，因此需要单独对 AP2 进行 Markov Chain 分析，AP1 和 AP3 具有对称性，可以同时进行 Markov chain 分析。最后将两个 Markov chain 得到的关系联立计算系统中的三个 AP 的稳态概率。

本问题同样需要对系统总时间的构成进行分析，并根据时间占比计算信道吞吐量。

6.2 问题求解

6.2.1 数值求解

根据问题描述，绘制改进 Bianchi 模型的 Markov chain 如图 6.1 所示。

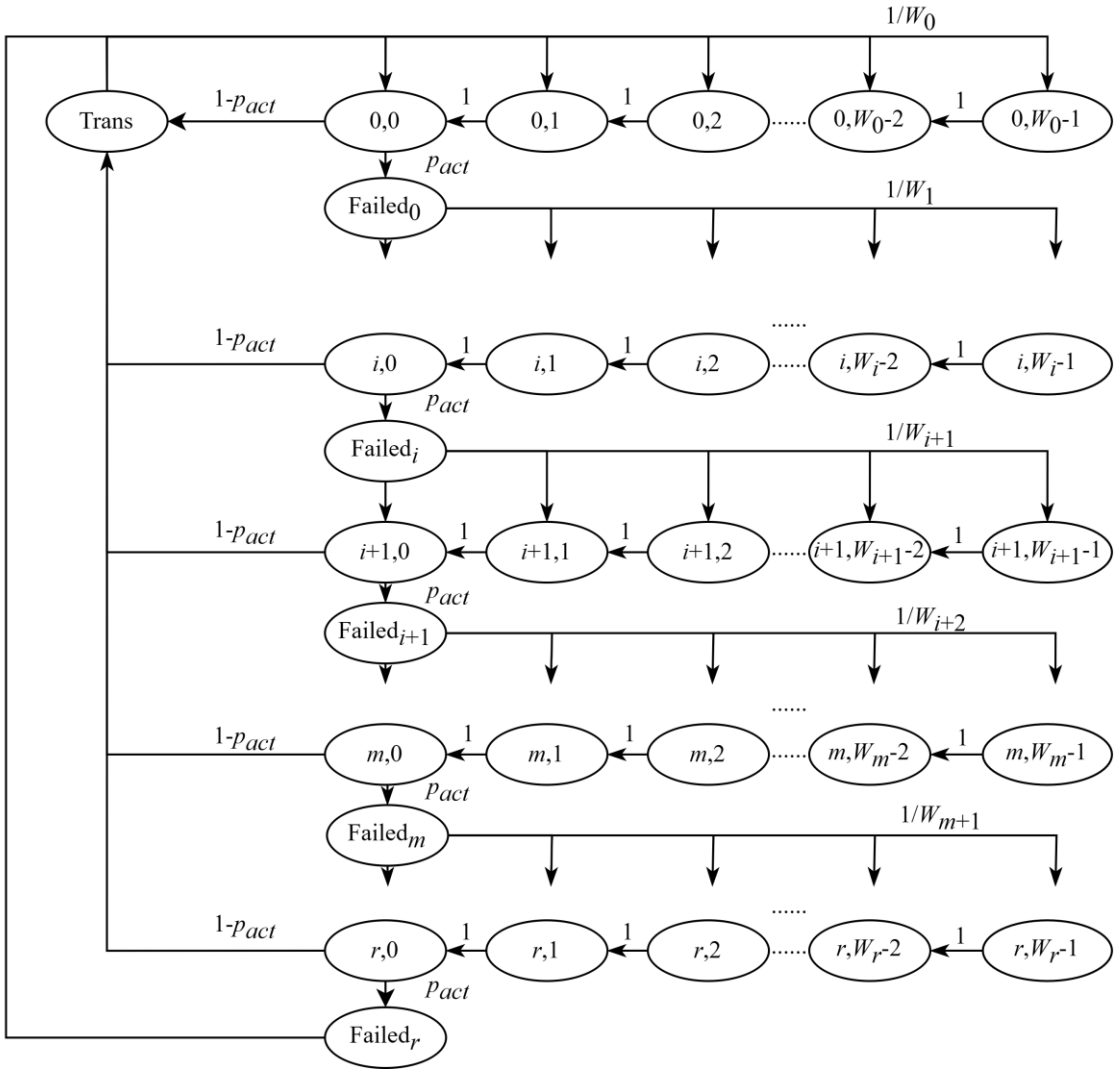


图 6.1 问题四 Markov chain

图 6.1 所示 Markov chain 状态转移概率通式如式（6.1）至式（6.5）所示。其中 p_{act} 表示节点的实际传输失败率。

$$P\{i,k\,|\,i,k+1\}=1, k\in[0,W_i-2], i\in[0,r] \tag{6.1}$$

$$P\{i,k\,|\,Failed_{i-1}\}=1/W_i, k\in[0,W_i-1], i\in[0,r] \tag{6.2}$$

$$P\{0,k\,|\,Trans\}=1/W_0, k\in[0,W_0-1], i\in[0,r] \tag{6.3}$$

$$P\{Trans\,|\,i,0\}=1-p_{act}, i\in[0,r] \tag{6.4}$$

$$P\{Failed_i\,|\,i,0\}=p_{act}, i\in[0,r] \tag{6.5}$$

记 $b_{i,k,1}$ 为 AP1 和 AP3 的 Markov chain 中的状态稳态概率，$b_{i,k,2}$ 为 AP2 的 Markov chain 中的状态稳态概率。Markov chain 各状态概率之和为 1，如式（6.6）所示。

$$\begin{aligned}1&=\sum_{i=0}^{r}\sum_{k=0}^{W_i-1}b_{i,k,m}+\sum_{i=0}^{r}p(Failed_i)+p(Trans)\\&=\sum_{i=0}^{r}\sum_{k=0}^{W_i-1}\frac{W_i-k}{W_i}b_{i,0,m}+\sum_{i=0}^{r}b_{i,0,m}\\&=\sum_{i=0}^{r}b_{i,0,m}\cdot\frac{W_i+3}{2},\qquad m\in\{1,2\}\end{aligned}\tag{6.6}$$

记 p_{act1} 表示 AP1 和 AP3 的实际失败传输条件概率， p_{act2} 表示 AP2 实际失败传输条件概率。对于不互听的 AP1 和 AP3，由问题三可知，其交叠条件概率 p_{ovelap} 可以由式（6.7）至式（6.12）得到。

$$\begin{aligned}p_1&=p\{Trans\}\cdot\frac{T_s-2(H+E(p))}{T_s}=(1-p_{act1})\sum_{i=0}^{r}b_{i,0}\cdot\frac{T_s-2(H+E(p))}{T_s}\\&=(1-p_{act1})\cdot\sum_{i=0}^{r}p_{act1}^{i}b_{0,0}\cdot\frac{T_s-2(H+E(p))}{T_s}\end{aligned}\tag{6.7}$$

$$p_2=p\{Trans\}\cdot\frac{H+E[P]}{T_s}\cdot\frac{2W_0-n}{2W_0}=(1-p_{act1})\sum_{i=0}^{r}p_{act1}^{i}b_{0,0}\cdot\frac{H+E[P]}{T_s}\cdot\frac{2W_0-n}{2W_0}\tag{6.8}$$

$$\begin{aligned}p_3&=p\{Failed\}\cdot\frac{T_c-2(H+E(p))}{T_c}=p_{act1}\sum_{i=0}^{r}b_{i,0}\cdot\frac{T_c-2(H+E(p))}{T_c}\\&=p_{act1}\cdot\sum_{i=0}^{r}p_{act1}^{i}b_{0,0}\cdot\frac{T_c-2(H+E(p))}{T_c}\end{aligned}\tag{6.9}$$

$$p_4=p\{Failed\}\cdot\frac{H+E[P]}{T_c}\cdot\frac{2W_i-n}{2W_i}=p_{act1}\sum_{i=0}^{r}p_{act1}^{i}b_{0,0}\cdot\frac{H+E[P]}{T_c}\cdot\frac{2W_i-n}{2W_i}\tag{6.10}$$

$$p_5=\sum_{i=0}^{r}\sum_{k=0}^{W_i-1}\gamma_k\cdot b_{i,k}=\sum_{i=0}^{r}\sum_{k=0}^{W_i-1}\gamma_k\cdot\frac{W_i-k}{W_i}\cdot b_{i,0}=\sum_{i=0}^{r}\sum_{k=0}^{W_i-1}\gamma_k\cdot\frac{W_i-k}{W_i}\cdot p_{act1}^{i}\cdot b_{0,0}\tag{6.11}$$

$$p_{overlap}=1-p_1-p_2-p_3-p_4-p_5\tag{6.12}$$

节点在一个时隙发送数据帧的概率为：

$$P(AP_1)=P(AP_3)=\tau_1=\sum_{i=0}^{r}b_{i,0,1}=b_{0,0,1}\cdot\frac{1-p_{act1}^{r+1}}{1-p_{act1}}\tag{6.13}$$

$$P(AP_2)=\tau_2=\sum_{i=0}^{r}b_{i,0,2}=b_{0,0,2}\cdot\frac{1-p_{act2}^{r+1}}{1-p_{act2}}\tag{6.14}$$

AP1 和 AP3 实际发送失败条件概率 p_{act1} 如式（6.15）所示。

$$p_{act1}=1-(1-\tau_2)\tag{6.15}$$

对 AP2，当 AP1、AP3 单独传输数据和交叠传输数据时，都会进入 hold time，因此其实际发送失败条件概率 p_{act2} 如式（6.16）所示。

$$\begin{aligned} P_{act2} &= P(AP_1 + AP_2) = P(AP_1) + P(AP_2) - P(AP_1AP_2) \\ &= 2\tau_1 - \tau_1 p_{overlap} \end{aligned} \tag{6.16}$$

联立式（6.6）、式（6.12）、式（6.13）、式（6.14）、式（6.15）和式（6.16）共 7 个方程，可分别解得 $b_{0,0,1}$、$b_{0,0,2}$、τ_1、τ_2、p_{act1}、p_{act2} 和 $p_{overlap}$。

该 3BSS 系统中时间构成如图所示。由于 AP1 和 AP3 不互听，因此这两个 AP 可以在另一 AP 传输数据时正常回退。AP2 则必须在另外两 AP 传输数据时进入 hold time。

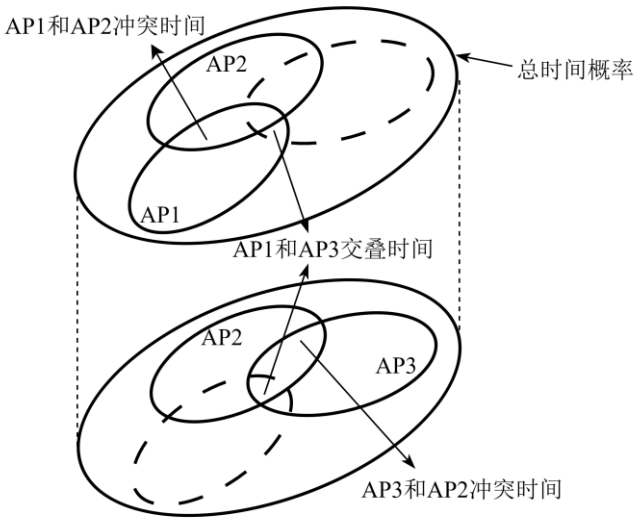


图 6.2 3BSS 系统时间构成

根据图 6.2 中 AP1 和 AP2 所在时间轴计算吞吐量。其中冲突导致的传输失败仅发生在 AP2 与另外两 AP 并行传输过程中，因此发生冲突的期望为：

$$E(Tc) = \tau_2 p_{act2} \times T_c \tag{6.17}$$

成功传输的期望和等待期望如式（6.18）和式（6.19）所示。

$$E(T_s) = [\tau_1(1 - p_{act1}) + \tau_2(1 - p_{act2})] \cdot T_s \tag{6.18}$$

$$E(T_e) = [1 - \tau_2 - \tau_1 \cdot (1 - p_{act1})]T_e \tag{6.19}$$

信道吞吐量计算如式（6.20）所示。

$$S = \frac{[2\tau_1(1 - p_{act1}) + \tau_2(1 - p_{act2})] \cdot E[P]}{E(T_s) + E(T_c) + E(T_e)} \cdot rate \tag{6.20}$$

本题计算得 $\tau_1 = 9.65\%$，$\tau_2 = 8.89\%$，$p_{overlap} = 42.13\%$，$p_{act1} = 8.89\%$，$p_{act2} = 15.23\%$，代入式（6.20）可得吞吐量 $S = 9.76 \times 10^7$ dps。

6.2.2 仿真检验

对于问题四的场景，根据 AP_a、AP_2 和 AP_b，$\forall(a,b) \in \{(1,3),(3,1)\}$ 谁先回退至 0 将离散事

件仿真模型划分为 5 种可能出现的情形，再根据 AP_a 和 AP_b 回退次数差异大小将情形 1 划分为情形 1-1 和情形 1-2，如图 6.3 所示。

(a) 在情形 1 中，令事件阶段为 p，AP_a 优先回退至 0，由于 AP_a 与 AP_b 两 BSS 不互听，AP_b 继续回退。在情形 1-1 中，由于 AP_a 传输完数据后 AP_b 仍在回退过程中，AP_a 和 AP_b 不发生交叠，AP_b 可能与下一事件阶段 $p+1$ 的 AP_a 发生交叠冲突，也可能成功传输，但均属于下一阶段需要考虑的问题，所以令 $newt_b = oldt_b$，即重置 AP_b 开始时间点，将下一事件阶段的 AP_a 与其进行比较，并重复此过程，直至 AP_b 进入下一事件阶段；由于 AP_a 和 AP_2 两者的 BSS 互听，当 AP_a 回退至 0 时，AP_2 尚未回退至 0，两者均同属一个事件阶段 p，因此 AP_a 成功传输。当 AP_2 回退完最后一个间隙 T_e 后，进入等待阶段直至 AP_a 传输完成。

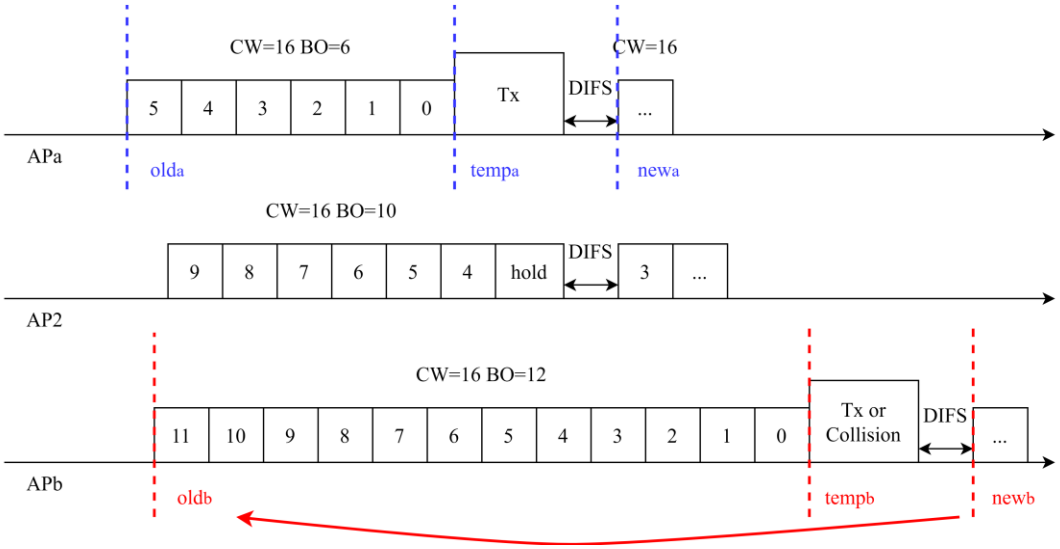
(b) 在情形 1-2 中，AP_a 数据未传输完成，AP_b 回退至 0，两者均同属一个事件阶段 p，由于 SIR 较大，AP_a 和 AP_b 均传输成功；AP_a 与 AP_b 不互听，AP_2 与两者都互听，当 AP_a 开始传输时，AP_2 尚未回退至 0，当 AP_2 等待时，AP_b 开始传输，AP_2 持续等待直至 AP_a 和 AP_b 均传输完成。

(c) 在情形 2 中，AP_2 优先回退至 0，由于 AP_a 与 AP_b 不互听，AP_2 与两者都互听，因此 AP_2 成功传输。当 AP_2 开始传输时，AP_a 与 AP_b 各自回退完一个间隙 T_e 后，均进入等待阶段直至 AP_2 传输完成。

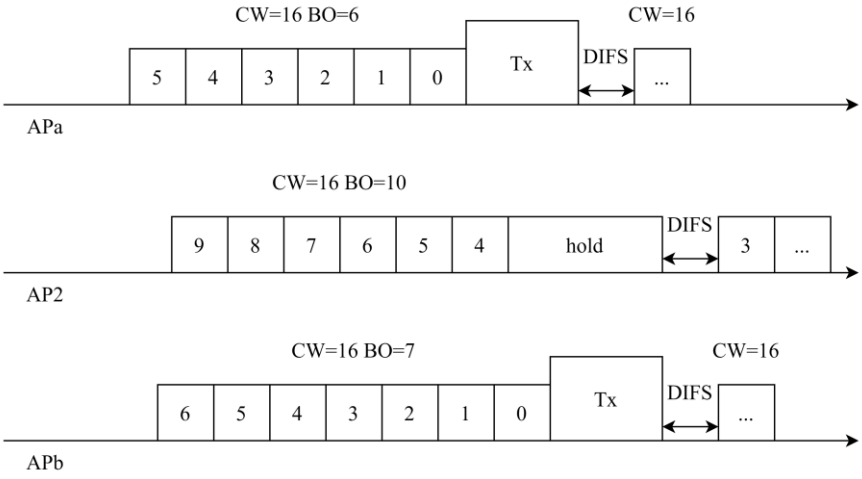
(d) 在情形 3 中，AP_a 和 AP_2 同时回退至 0，由于 AP_a 和 AP_2 两 BSS 互听，两者均传输失败；由于 AP_b 和 AP_2 互听，AP_b 回退完最后一个间隙 T_e 后，进入等待阶段直至冲突结束。

(e) 在情形 4 中，AP_a 和 AP_b 同时回退至 0，由于 AP_a 与 AP_b 两 BSS 不互听，两者均能成功传输；由于 AP_2 与两者均互听，AP_2 回退完最后一个间隙 T_e 后，进入等待阶段直至传输完成。

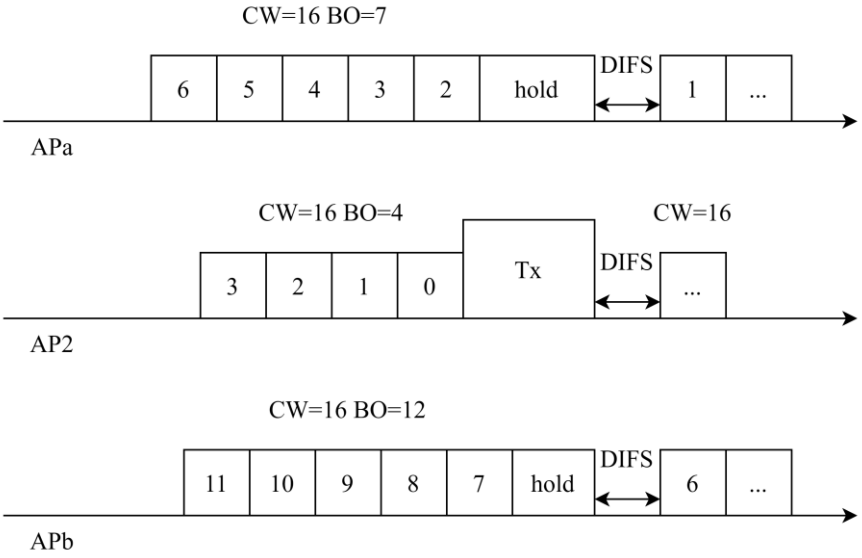
(f) 在情形 5 中，AP_a、AP_2 和 AP_b 同时回退至 0，由于 AP_a 与 AP_b 不互听，AP_2 与两者都互听，发生冲突，三者均传输失败。



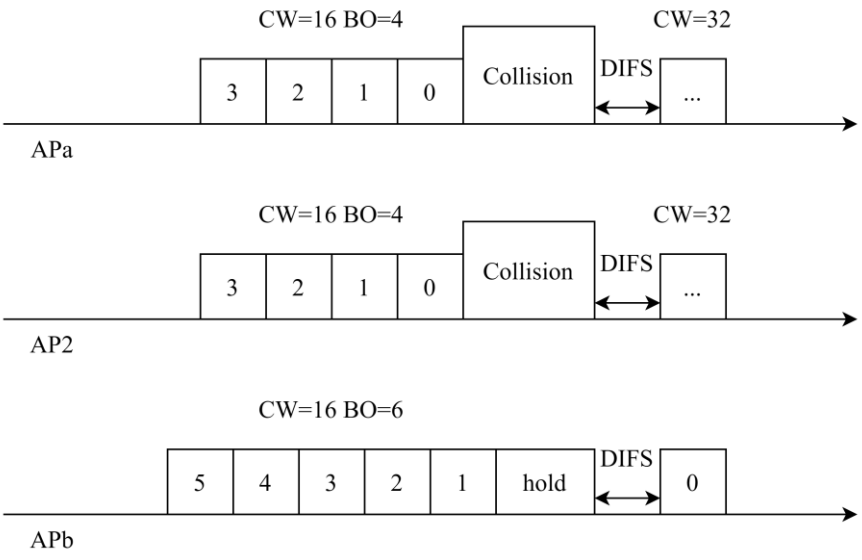
（a）


CW=16 BO=6
CW=16
5
4
3
2
1
0
Tx
DIFS
...
APa
CW=16 BO=10
9
8
7
6
5
4
hold
DIFS
3
...
AP2
CW=16 BO=7
CW=16
6
5
4
3
2
1
0
Tx
DIFS
...
APb

（b）


CW=16 BO=7
6
5
4
3
2
hold
DIFS
1
...
APa
CW=16 BO=4
CW=16
3
2
1
0
Tx
DIFS
...
AP2
CW=16 BO=12
11
10
9
8
7
hold
DIFS
6
...
APb

（c）


CW=16 BO=4
CW=32
3
2
1
0
Collision
DIFS
...
APa
CW=16 BO=4
CW=32
3
2
1
0
Collision
DIFS
...
AP2
CW=16 BO=6
5
4
3
2
1
hold
DIFS
0
APb

（d）

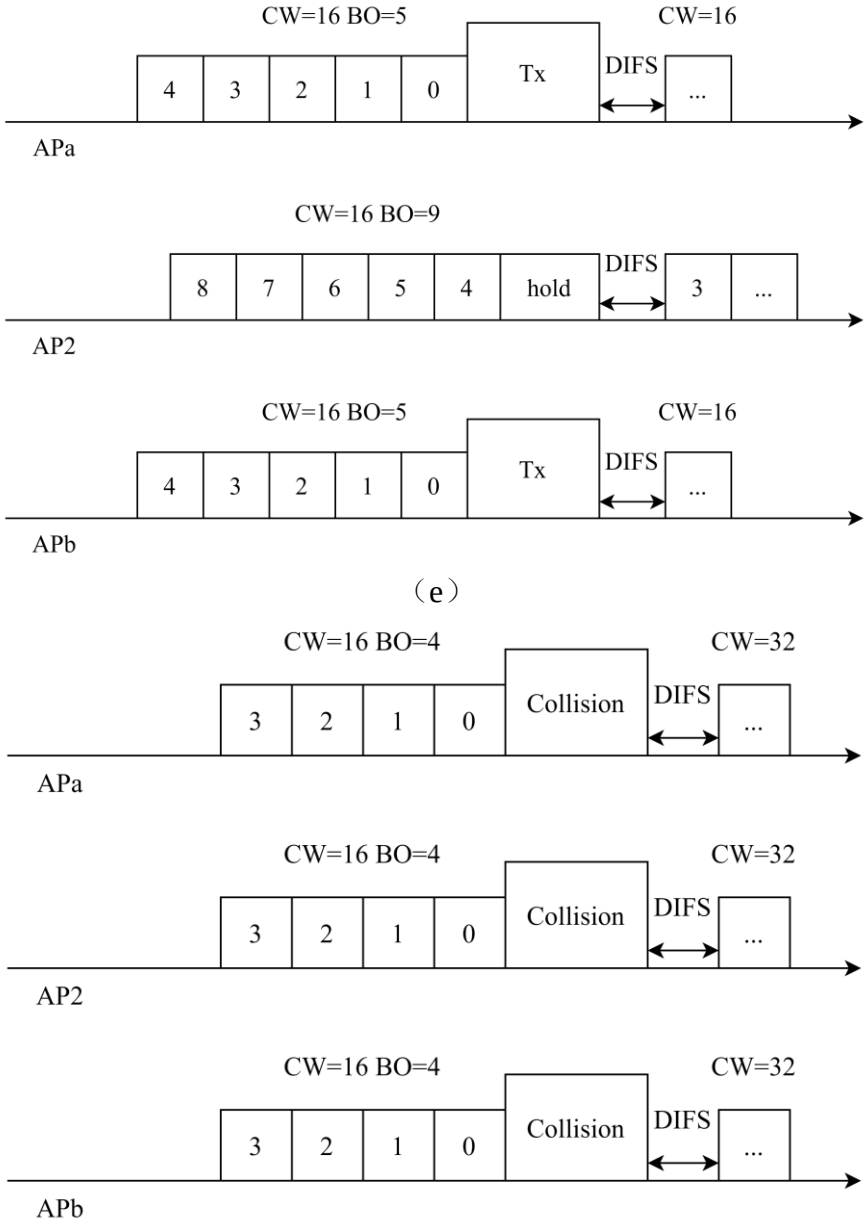


图 6.3 问题四情形示意图（a）情形 1-1 （b）情形 1-2 （c）情形 2 （d）情形 3 （e）情形 4 （f）情形 5

问题四实验结果如表 6.1 所示，统计计算结果如图 6.4 所示，均值为 $\overline{S}=1.1\times10^{8}$dps，标准差为$\sigma=3.5\times10^{4}$。仿真结果与改进 Bianchi 模型计算得到的结果相差 12.27%，证明了 Bianchi 模型的准确性。仿真结果概率图如图 6.5 所示，使用 Minitab 计算出 P 值为 0.767 大于 0.05，实验结果服从正态分布。

表 6.1 问题四仿真器实验结果

单位：dps

序号	实验结果	序号	实验结果	序号	实验结果	序号	实验结果
1	110189047	11	110161106	21	110145573	31	110200219
2	110218369	12	110162998	22	110151403	32	110126427

续表 6.1

序号	实验结果	序号	实验结果	序号	实验结果	序号	实验结果
3	110163517	13	110161312	23	110127523	33	110147504
4	110153522	14	110109577	24	110187915	34	110234720
5	110183703	15	110140386	25	110130020	35	110166142
6	110143319	16	110264378	26	110159257	36	110171757
7	110191848	17	110194872	27	110079578	37	110174729
8	110184929	18	110222367	28	110171122	38	110188984
9	110146376	19	110200330	29	110189165	39	110205864
10	110154648	20	110176632	30	110196391	40	110195468

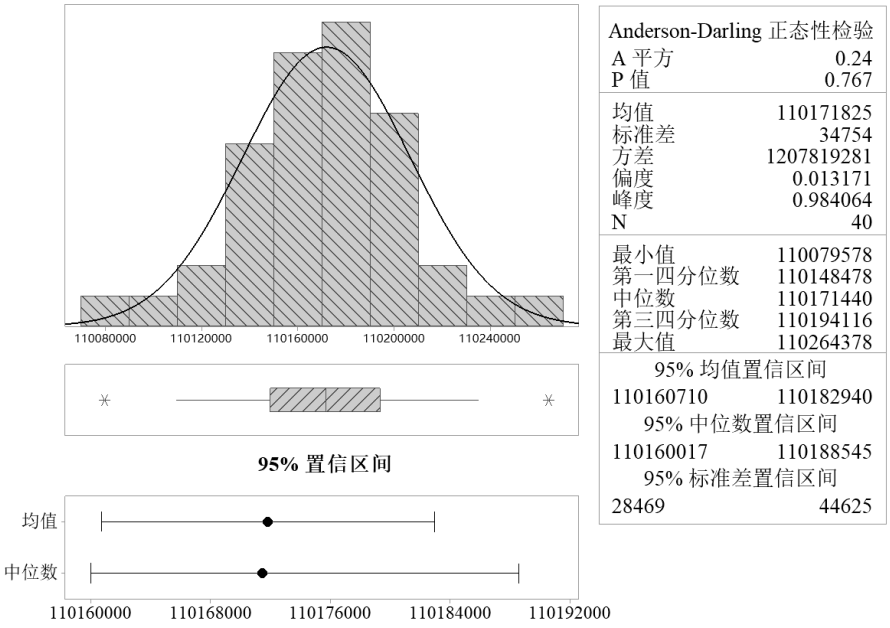


图 6.4 问题四仿真结果正态性检验报告

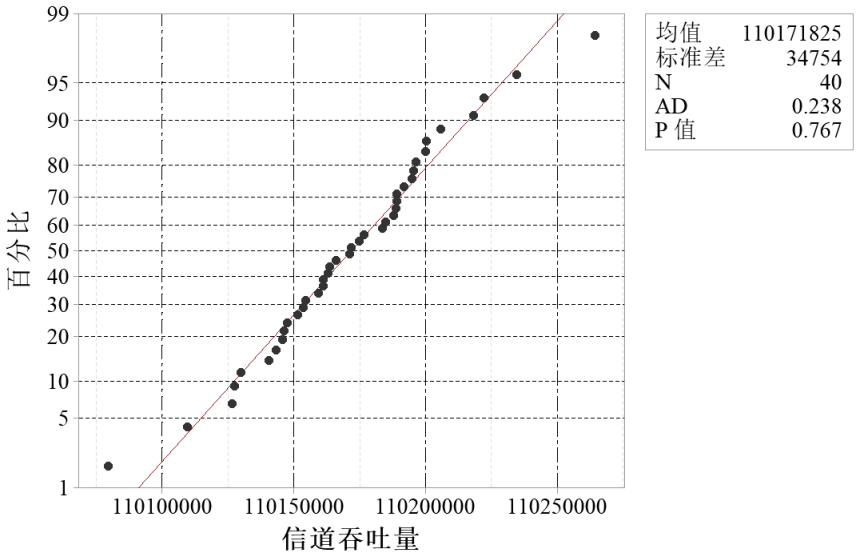


图 6.5 问题四仿真结果概率图

当系统取其他参数时，Bianchi 模型和仿真实验计算结果如表 6.2 所示，数值计算和仿真实验的误差控制在 5%-15%之间，证明了改进 Bianchi 模型的准确性。系统取其他参数时的仿真结果正态性检验概率图见附录 B。

表 6.2 其他参数下两方法结果对比

吞吐量单位：bps

序号	CW_{min}	CW_{max}	r	物理层速率	数值计算	仿真实验	Gap
1	16	1024	32	455.8Mbps	89729360	101909404	11.95%
2	16	1024	6	286.8Mbps	77076783	81048402	4.90%
3	32	1024	5	286.8Mbps	89729857	101916034	11.96%
4	16	1024	32	286.8Mbps	76234106	87737732	13.11%
5	16	1024	6	158.4Mbps	67038178	71161702	5.79%
6	32	1024	5	158.4Mbps	76234207	87736668	13.11%

7 模型评价

7.1 优点

(1) 根据离散事件仿真思想对 DSMA/CA 机制在不同场景进行建模仿真模拟，依靠计算机能够获得较为贴近实际的仿真过程，并得到较为可靠的信道吞吐量评估结果。

(2) 对传统的 Bianchi 模型进行了创新，增加了信号包成功传输状态 Trans 和信号包传输失败状态 Failed，该改进使得传统模型更加贴近于真实场景的机制，并与离散事件仿真结果进行对比，在某些场景下能够比传统 Bianchi 模型有更高的精确度。

7.2 不足

本文的数值分析方法局限于对 Bianchi 模型的研究和改进，该传统模型在理想的假设下有简洁易用的优势，但当模型越发复杂时，建模思路变得复杂和困难，不同的参数设置也会使其估算结果与仿真结果不存在稳定跟随的关系。采用除 Bianchi 模型以外的其他模型框架进行数值分析，可能会得到更精确的解。

7.3 展望

本文对设定了一些理想情况下的 CSMA/CA 机制的信道接入场景进行了建模和分析，当理想假设减少后，建立的模型将会更加贴近于真实的网络通讯情景。对更加贴近现实的模型进行建立，有助于加深对现实通讯技术的认识并提出对现有通讯质量技术进一步发展的可行性方案，未来可研究包括多 BSS 物理速率不同等网络通讯情景，并挖掘其内部机理。WLAN 技术的进步会推动万物互联，虚拟现实等益于提高人类生活质量技术的进步。

参考文献

[1] Hanzo L, Münster M, Choi B J. OFDM and MC-CDMA for Broadband Multi-User Communications, WLANs and Broadcasting[J]. John Wiley & Sons, Inc. 2003.

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

[3] 杜进. 面向工业场景的 Wi-Fi 网络 AP 优化部署与信道优化分配研究[D]. 东南大学, 2022.

[4] 郁帅鑫. 智能家居场景下 WLAN 随机接入优化研究[D]. 华中科技大学, 2022.

[5] 彭林哲. 无线局域网 DCF 性能分析与公平性改进研究[D]. 湖南大学, 2017.

[6] Kleijnen J P C, Bettonvil B, Persson F. Finding the Important Factors in Large Discrete-Event Simulation: Sequential Bifurcation and its Applications[J]. Social Science Electronic Publishing, 2009, 2003-104(2003-104): 287-307.

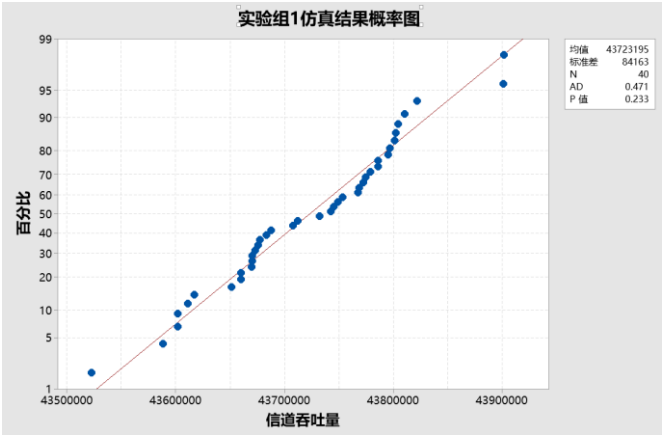
[7] Jafer S, Liu Q, Wainer G. Synchronization methods in parallel and distributed discrete-event simulation[J]. Simulation Modelling Practice and Theory, 2013, 30: 54-73.

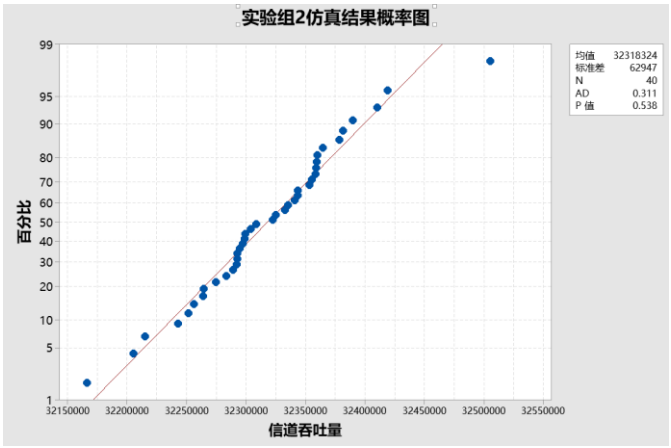
[8] 崔晓峰. 面向对象的离散事件仿真建模与实现[J]. 计算机工程与应用, 1998(09): 40-41+47.

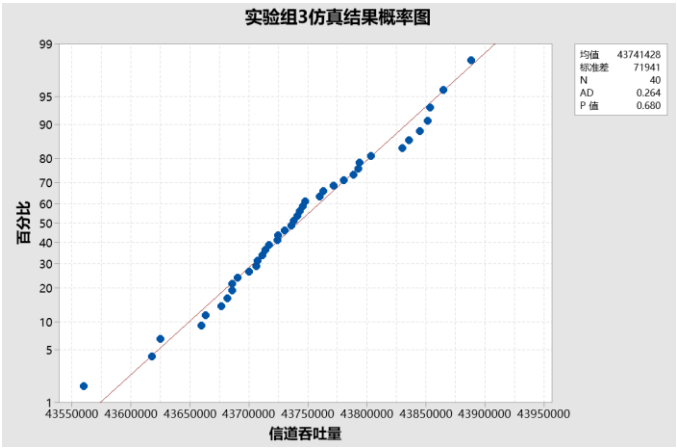
[9] 刘勇, 王德才, 冯正超. 离散事件系统仿真建模与仿真策略[J]. 西南师范大学学报(自然科学版), 2005(06): 1019-1025.

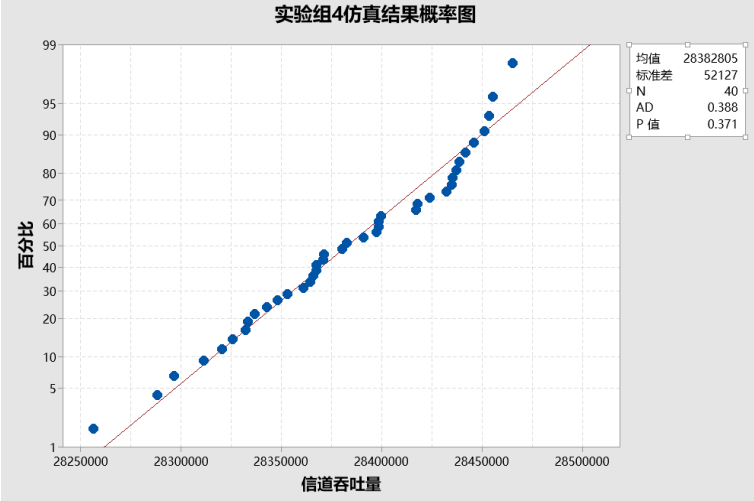
附录 A 问题三其他参数下仿真结果正态性检验概率图

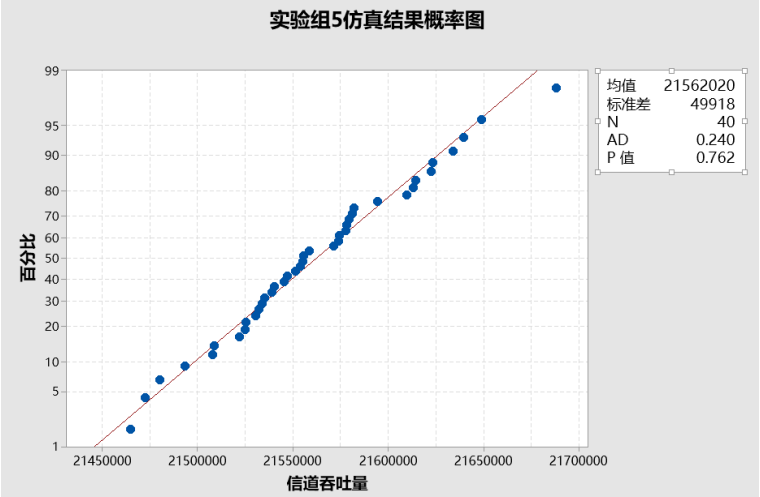
实验组号	CW_{min}	CW_{max}	r	物理层速率
1	16	1024	32	455.8Mbps
2	16	1024	6	286.8Mbps
3	32	1024	5	286.8Mbps
4	16	1024	32	286.8Mbps
5	16	1024	6	158.4Mbps
6	32	1024	5	158.4Mbps

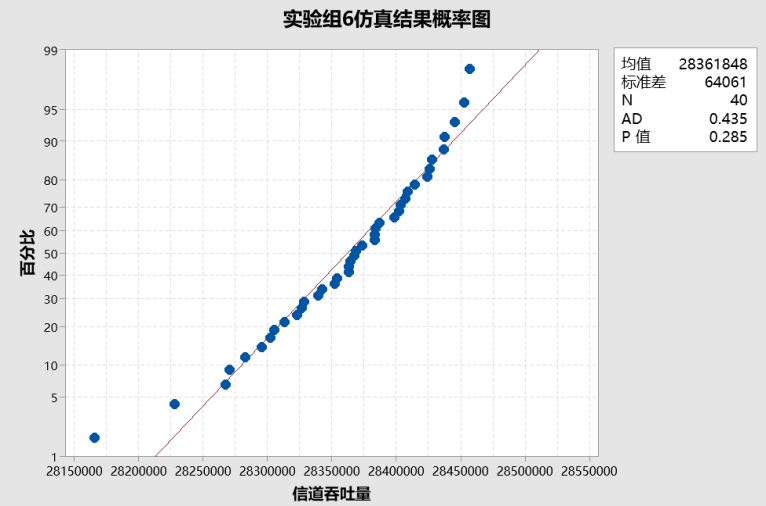




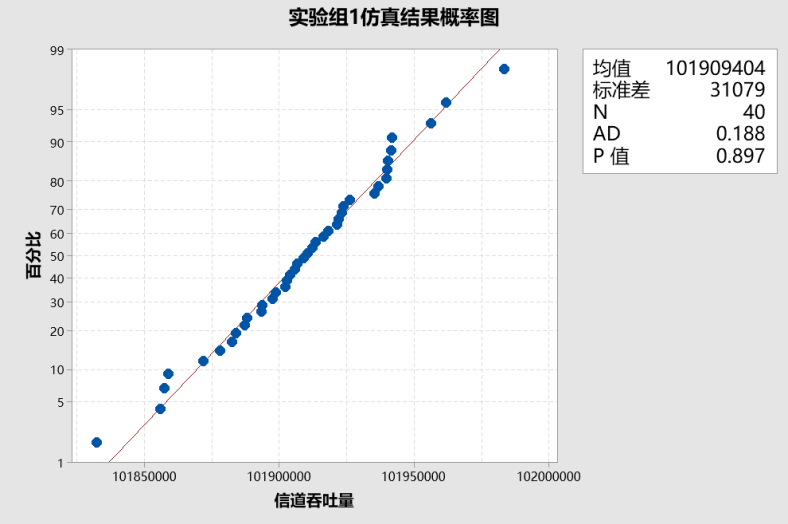


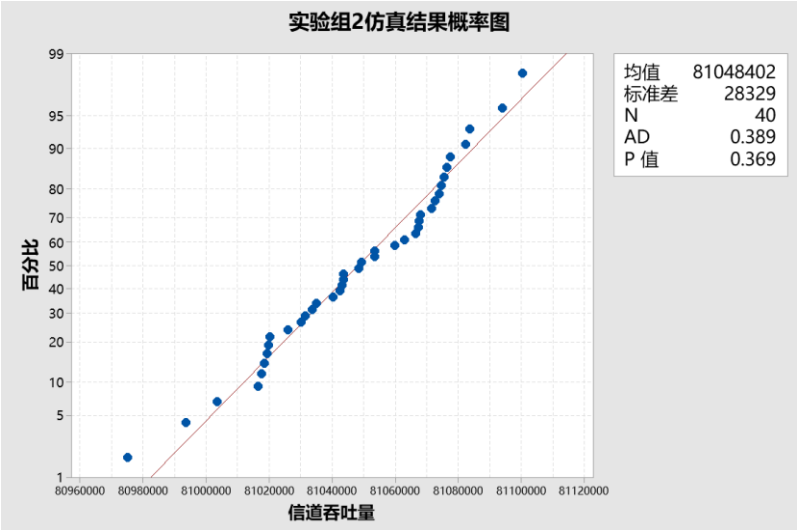

实验组4仿真结果概率图
均值 28382805
标准差 52127
N 40
AD 0.388
P 值 0.371
百分比
99
95
90
80
70
60
50
40
30
20
10
5
1
28250000
28300000
28350000
28400000
28450000
28500000
信道吞吐量

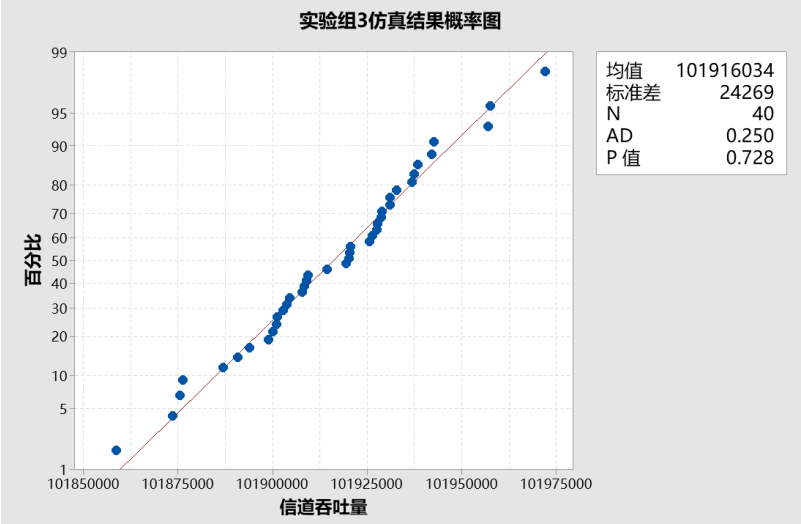

实验组5仿真结果概率图
均值 21562020
标准差 49918
N 40
AD 0.240
P 值 0.762
百分比
99
95
90
80
70
60
50
40
30
20
10
5
1
21450000
21500000
21550000
21600000
21650000
21700000
信道吞吐量

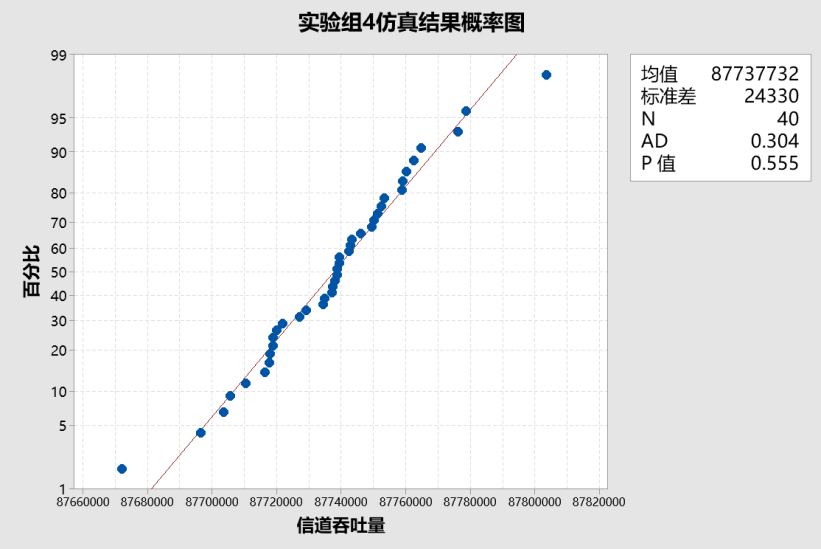

实验组6仿真结果概率图
均值 28361848
标准差 64061
N 40
AD 0.435
P 值 0.285
百分比
99
95
90
80
70
60
50
40
30
20
10
5
1
28150000
28200000
28250000
28300000
28350000
28400000
28450000
28500000
28550000
信道吞吐量

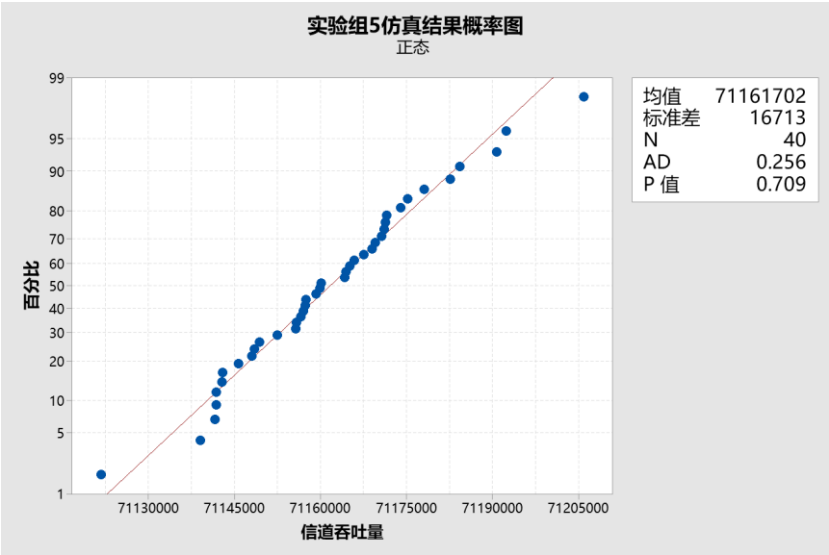
附录 B 问题四其他参数下仿真结果正态性检验概率图

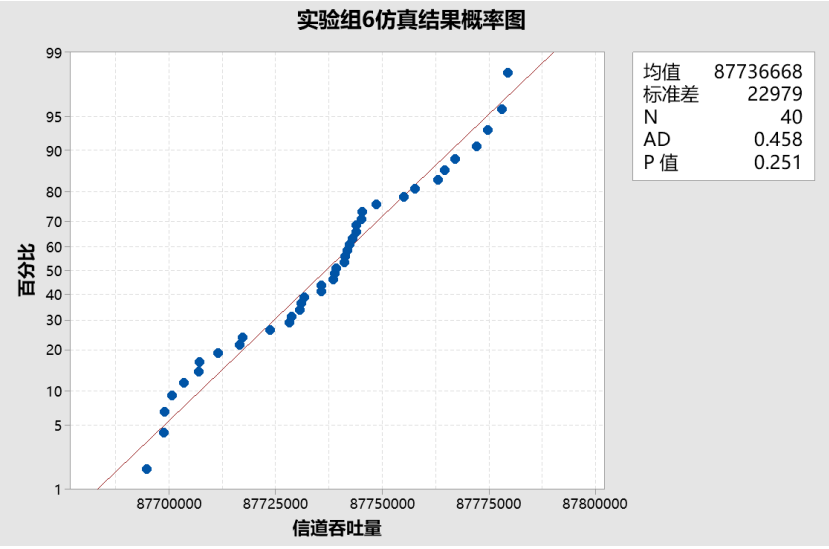







实验组4仿真结果概率图
99
95
90
80
70
60
50
40
30
20
10
5
1
百分比
87660000
87680000
87700000
87720000
87740000
87760000
87780000
87800000
87820000
信道吞吐量
均值 87737732
标准差 24330
N 40
AD 0.304
P 值 0.555


实验组5仿真结果概率图
正态
99
95
90
80
70
60
50
40
30
20
10
5
1
百分比
71130000
71145000
71160000
71175000
71190000
71205000
信道吞吐量
均值 71161702
标准差 16713
N 40
AD 0.256
P 值 0.709


实验组6仿真结果概率图
99
95
90
80
70
60
50
40
30
20
10
5
1
百分比
87700000
87725000
87750000
87775000
87800000
信道吞吐量
均值 87736668
标准差 22979
N 40
AD 0.458
P 值 0.251

附录 C 代码

问题一数值求解 Matlab 代码：

标准 Bianchi：

```
function F = main1(x)
w0 = 16;
wmax = 1024;
r = 0;
rate = 455.8;
m = log(wmax/w0)/log(2);
w = zeros(1,r+1);
EP = 1500/rate/125000*1000000;
for i = 0:r
    if (i<=m)
        w(i+1) = 2^i*w0;
    elseif (i<=r)
        w(i+1) = 2^m*w0;
    end
end
F(1) = x(1) * (1-x(2)^(r+1))/(1-x(2)) - x(2);
F(2) = - x(1) + 2 * (1-x(2)) * (1-2*x(2)) / ( w0*(1-(2*x(2))^(m+1))*(1-x(2)) +
(1-2*x(2))*(1-x(2)^(r+1)) + w0*2^m*x(2)^(m+1)*(1-x(2)^(r-m))*(1-2*x(2)));
F(3) = x(3) - x(2);
```

改进 Bianchi main10.mat

```
function F = main10(x)
w0 = 16;
wmax = 1024;
r = 32;
rate = 455.8;
m = log(wmax/w0)/log(2);
w = zeros(1,r+1);
EP = 1500/rate/125000*1000000;
d = 30/rate/125000*1000000 + 13.6 + EP;
for i = 0:r
    if (i<=m)
        w(i+1) = 2^i*w0;
    elseif (i<=r)
        w(i+1) = 2^m*w0;
    end
end
```

```
sum = 0;
for i = 0:r
    sum = sum + x(2)^i * (w(i+1) + 3) / 2;
end
dd = 0;
for i = 0:r
    dd = dd + x(2)^i * x(1);
end
F(1) = dd - x(3);
F(2) = x(1) * sum - 1;
F(3) = x(2)-x(3);
```

求解吞吐量 solve1.m

```
fun = @main10;
% fun = @main1
x0 = [0,0,0];
x = fsolve(fun,x0)
t = x(3);
p = x(2);
w0 = 16;
wmax = 1024;
r = 32;
rate = 455.8;
m = log(wmax/w0)/log(2);
w = zeros(1,r+1);
EP = 1500/rate/125000*1000000;
d = 30/rate/125000*1000000 + 13.6 + EP;
Ts = 30/rate/125000*1000000 + 13.6 + EP + 16 + 32 + 43;
Tc = 30/rate/125000*1000000 + 13.6 + EP + 43 + 65;
e = 9;
S = 2*t*(1-p)*EP*rate*1000000/(2*t*(1-p)*Ts+t*p*Tc+(1-2*t+t*p)*e)
```

问题一仿真求解 python 代码：

```
import random
import xlwt
from math import log
CWmin = 16
CWmax = 1024
r = 32 #最大重传次数
rate = 455.8
Te = 9
EP = 1500/rate/125000*1000000
```

```
Ts = 30/rate/125000*1000000 + 13.6 + EP + 16 + 32 + 43
Tc = 30/rate/125000*1000000 + 13.6 + EP + 43 + 65
PMP = 30/rate/125000*1000000 + 13.6 + EP  #PHY hdr + MAC hdr + Payload
SIFS = 16
DIFS = 43
ACK = 32
ACktimeout = 65
m = log(CWmax/CWmin,2)
#计算数据分布
data_list = []
for i in range(40):
    time = DIFS #记录时间
    cw1icounter, cw2icounter = 0, 0 #AP 数据的发送次数
    cw1jcounter, cw2jcounter = 0, 0 #AP 回退翻倍次数
    collisioncounter = 0 #崩溃次数
    cw1rollbackflag, cw2rollbackflag = 1, 1 #判断是否需要产生新的回退
    #离散事件仿真
    for j in range(10**6):
        if cw1icounter <= m:
            cw1jcounter = cw1icounter
        elif m < cw1icounter < r: #超过最大退避阶数
            cw1jcounter = m
        elif cw1icounter >= r: #超过最大重传次数
            cw1icounter, cw1jcounter = 0, 0
        else:
            pass
        if cw2icounter <= m:
            cw2jcounter = cw2icounter
        elif m < cw2icounter < r:
            cw2jcounter = m
        elif cw2icounter >= r:
            cw2icounter, cw2jcounter = 0, 0
        else:
            pass
        if cw1rollbackflag == 1:
            CW1 = random.randint(0,(CWmin*(2**cw1jcounter)-1))
        else:
            pass
        if cw2rollbackflag == 1:
            CW2 = random.randint(0,(CWmin*(2**cw2jcounter)-1))
        else:
            pass
        if CW1 < CW2:
```

```
            cw1rollbackflag, cw2rollbackflag = 1, 0
            time = time + (CW1+1)*Te + Ts
            CW2 = CW2 - CW1 - 1
            cw1icounter = 0 #成功上传重置 CW
        elif CW2 < CW1:
            cw1rollbackflag, cw2rollbackflag = 0, 1
            time = time + (CW2+1)*Te + Ts
            CW1 = CW1 - CW2 - 1
            cw2icounter = 0
        else: #同时崩溃
            cw1rollbackflag, cw2rollbackflag = 1, 1 #冲突产生新的 CW
            cw1icounter = cw1icounter + 1 #冲突更新数据发送次数
            cw2icounter = cw2icounter + 1
            collisioncounter = collisioncounter + 1
            time = time + (CW1+1)*Te + Tc
    data = (10**6 - collisioncounter)*EP*rate*(10**6)/time
    data_list.append(data)
#excel
work_book = xlwt.Workbook(encoding = 'utf-8')
work_sheet = work_book.add_sheet('sheet')
for i in range(len(data_list)):
    work_sheet.write(i, 0, data_list[i])
work_book.save('experiment_q1.xls')
```

问题二仿真求解 python 代码：

```
import random
import xlwt
from math import log
CWmin = 16
CWmax = 1024
r = 32 #最大重传次数
rate = 275.3
Te = 9
EP = 1500/rate/125000*1000000
Ts = 30/rate/125000*1000000 + 13.6 + EP + 16 + 32 + 43
Tc = 30/rate/125000*1000000 + 13.6 + EP + 43 + 65
PMP = 30/rate/125000*1000000 + 13.6 + EP  #PHY hdr + MAC hdr + Payload
SIFS = 16
DIFS = 43
ACK = 32
ACktimeout = 65
m = log(CWmax/CWmin,2)
```

```
#计算数据分布
data_list = []
for i in range(40):
    time = DIFS #记录时间
    cw1counter, cw2counter = 0, 0 #记录成功传输次数
    cw1rollbackflag, cw2rollbackflag = 1, 1 #判断是否需要产生新的回退
    #离散事件仿真
    for j in range(10**6):
        if cw1rollbackflag == 1:
            CW1 = random.randint(0,CWmin-1)
        else:
            pass
        if cw2rollbackflag == 1:
            CW2 = random.randint(0,CWmin-1)
        else:
            pass
        if CW1 < CW2:
            cw1rollbackflag, cw2rollbackflag = 1, 0
            time = time + (CW1+1)*Te + Ts
            CW2 = CW2 - CW1 - 1
            cw1counter = cw1counter + 1
        elif CW2 < CW1:
            cw1rollbackflag, cw2rollbackflag = 0, 1
            time = time + (CW2+1)*Te + Ts
            CW1 = CW1 - CW2 - 1
            cw2counter = cw2counter + 1
        else: #同时上传
            cw1rollbackflag, cw2rollbackflag = 1, 1
            time = time + (CW1+1)*Te + Ts
            cw1counter = cw1counter + 1
            cw2counter = cw2counter + 1
    data = 2*cw1counter*EP*rate*(10**6)/time
    data_list.append(data)
#excel
work_book = xlwt.Workbook(encoding = 'utf-8')
work_sheet = work_book.add_sheet('sheet')
for i in range(len(data_list)):
    work_sheet.write(i, 0, data_list[i])
work_book.save('experiment_q2.xls')
```

问题三数值求解 Matlab 代码：

Bianchi 模型 main3.m

```
function F = main3(x)
w0 = 16;
wmax = 1024;
r = 32;
rate = 455.8;
m = log(wmax/w0)/log(2);
w = zeros(1,r+1);
EP = 1500/rate/125000*1000000;
d = 30/rate/125000*1000000 + 13.6 + EP;
Ts = 30/rate/125000*1000000 + 13.6 + EP + 16 + 32 + 43;
Tc = 30/rate/125000*1000000 + 13.6 + EP + 43 + 65;
e = 9;
kk = d/e;
for i = 0:r
    if (i<=m)
        w(i+1) = 2^i*w0;
    elseif (i<=r)
        w(i+1) = 2^m*w0;
    end
end
sum = 0;
for i = 0:r
    sum = sum + x(2)^i * (w(i+1) + 3) / 2;
end
s = 0;
for i = 0:r
    s = s + (1-x(2)) * x(2)^i * x(1) * (Ts-2*d)/Ts;
    s = s + x(2) * x(2)^i * x(1) * (Tc-2*d)/Tc;
    s = s + (w0-kk+w0)/w0/2 * (1-x(2)) * x(2)^i * x(1) * (d)/Ts;
    s = s + (w(i+1)-kk+w(i+1))/w(i+1)/2 * x(2) * x(2)^i * x(1) * (d)/Tc;
    k = ceil(kk);
    s = s + (w(i+1)-k+1)*(w(i+1)-k)/2/w(i+1) * x(2)^i * x(1);
    l = floor(kk);
    a = 1 - (kk * 100- l*100)/100;
    s = s + a * (w(i+1)-l)/w(i+1) * x(2)^i * x(1);
end
dd = 0;
for i = 0:r
    dd = dd + x(2)^i * x(1);
end
F(1) = dd - x(3);
% F(1) = x(1) * (1-x(2)^(r+1))/(1-x(2)) - x(3);
F(2) = x(1) * sum - 1;
```

```
F(3) =  1 - s - x(4);%%%%%%%2
% F(3) = x(2)-x(3); %%%%%%%%1
% F(4) = x(4) + (1-x(4)) * 0.1 - x(2);%%%%%%%2
F(4) = x(3) - x(2);
```

求解 solve3.m

```
w0 = 16;
wmax = 1024;
r = 32;
rate = 455.8;
m = log(wmax/w0)/log(2);
w = zeros(1,r+1);
EP = 1500/rate/125000*1000000;
d = 30/rate/125000*1000000 + 13.6 + EP;
Ts = 30/rate/125000*1000000 + 13.6 + EP + 16 + 32 + 43;
Tc = 30/rate/125000*1000000 + 13.6 + EP + 43 + 65;
e = 9;
kk = d/e;
fun = @main3;
x0 = [0.5 ,0.5 ,0.5,0];
x = fsolve(fun,x0)
t = x(3);
% p = x(2);   %p13
p = x(4);    %p3
pe = 0.1;
% S = 2*t*(1-p)*EP*rate*1000000/(2*t*(1-p)*Ts+t*p*Tc+(1-2*t+t*p)*e)
% S = 2*t*EP*rate*1000000/(2*t*(1-p)*Ts+t*p*Ts+(1-2*t+t*p)*e)
% S = (2*t*(1-p-pe)*EP)/((1-t)*e+t*(1-p-pe)*Ts+t*(pe+p)*Tc)*rate*1000000
```

问题三仿真求解 python 代码：

```
import random
import xlwt
from math import log
CWmin = 16
CWmax = 1024
r = 32 #最大重传次数
rate = 158.4
Te = 9
EP = 1500/rate/125000*1000000
Ts = 30/rate/125000*1000000 + 13.6 + EP + 16 + 32 + 43
Tc = 30/rate/125000*1000000 + 13.6 + EP + 43 + 65
PMP = 30/rate/125000*1000000 + 13.6 + EP  #PHY hdr + MAC hdr + Payload
```

```
SIFS = 16
DIFS = 43
ACK = 32
ACktimeout = 65
m = log(CWmax/CWmin,2)
#计算数据分布
data_list = []
for i in range(40):
    time1_old, time2_old = DIFS, DIFS #记录时间
    cw1icounter, cw2icounter = 0, 0 #AP 数据的发送次数
    cw1jcounter, cw2jcounter = 0, 0 #AP 回退翻倍次数
    cw1rollbackflag, cw2rollbackflag = 1, 1 #判断是否需要产生新的回退
    AP1successcounter = 0 #AP1 成功传输次数计数器
    AP1flag = 0 #判别 AP1 是否成功
    #离散事件仿真
    for j in range(10**6):
        if cw1icounter <= m:
            cw1jcounter = cw1icounter
        elif m < cw1icounter < r: #超过最大退避阶数
            cw1jcounter = m
        elif cw1icounter >= r: #超过最大重传次数
            cw1icounter, cw1jcounter = 0, 0
        else:
            pass
        if cw2icounter <= m:
            cw2jcounter = cw2icounter
        elif m < cw2icounter < r:
            cw2jcounter = m
        elif cw2icounter >= r:
            cw2icounter, cw2jcounter = 0, 0
        else:
            pass
        if cw1rollbackflag == 1:
            CW1 = random.randint(0,(CWmin*(2**cw1jcounter)-1))
        else:
            pass
        if cw2rollbackflag == 1:
            CW2 = random.randint(0,(CWmin*(2**cw2jcounter)-1))
        else:
            pass
        time1_temp = time1_old + CW1*Te
        time2_temp = time2_old + CW2*Te
        if time1_temp > time2_temp: #站点 2 优先回退至 0
```

```
    if time2_temp + PMP <= time1_temp: #站点 2 信号在站点 1 回退至 0 前不发生重叠
        if random.random() <= 0.9: #不丢包
            time1_new = time1_temp + PMP + SIFS + ACK + DIFS
            AP1successcounter = AP1successcounter + 1
            AP1flag = 1
            cw1icounter = 0 #成功上传重置 CW
        else: #丢包概率 10%
            time1_new = time1_temp + PMP + ACktimeout + DIFS
            AP1flag = 0
            cw1icounter = cw1icounter + 1
        if random.random() <= 0.9: #不丢包
            time2_new = time2_temp + PMP + SIFS + ACK + DIFS
            cw2icounter = 0 #成功上传重置 CW
        else: #丢包概率 10%
            time2_new = time2_temp + PMP + ACktimeout + DIFS
            cw2icounter = cw2icounter + 1
    else: #站点 2 信号在站点 1 回退至 0 前发生重叠
        time1_new = time1_temp + PMP + ACktimeout + DIFS
        AP1flag = 0
        time2_new = time2_temp + PMP + ACktimeout + DIFS
        cw1icounter = cw1icounter + 1
        cw2icounter = cw2icounter + 1
elif time1_temp < time2_temp: #站点 1 优先回退至 0
    if time1_temp + PMP <= time2_temp: #站点 1 信号在站点 2 回退至 0 前不发生重叠
        if random.random() <= 0.9: #不丢包
            time1_new = time1_temp + PMP + SIFS + ACK + DIFS
            AP1successcounter = AP1successcounter + 1
            AP1flag = 1
            cw1icounter = 0 #成功上传重置 CW
        else: #丢包概率 10%
            time1_new = time1_temp + PMP + ACktimeout + DIFS
            AP1flag = 0
            cw1icounter = cw1icounter + 1
        if random.random() <= 0.9: #不丢包
            time2_new = time2_temp + PMP + SIFS + ACK + DIFS
            cw2icounter = 0 #成功上传重置 CW
        else: #丢包概率 10%
            time2_new = time2_temp + PMP + ACktimeout + DIFS
            cw2icounter = cw2icounter + 1
    else: #站点 1 信号在站点 2 回退至 0 前发生重叠
        time1_new = time1_temp + PMP + ACktimeout + DIFS
        AP1flag = 0
        time2_new = time2_temp + PMP + ACktimeout + DIFS
```

```
            cw1icounter = cw1icounter + 1
            cw2icounter = cw2icounter + 1
        else: #同时回退至 0 肯定发生重叠
            time1_new = time1_temp + PMP + ACktimeout + DIFS
            AP1flag = 0
            time2_new = time2_temp + PMP + ACktimeout + DIFS
            cw1icounter = cw1icounter + 1
            cw2icounter = cw2icounter + 1
        if time1_new <= time2_temp:
            cw1rollbackflag, cw2rollbackflag = 1, 0
            time2_new = time2_old
        else:
            cw1rollbackflag, cw2rollbackflag = 1, 1
        if time2_new <= time1_temp:
            cw1rollbackflag, cw2rollbackflag = 0, 1
            time1_new = time1_old
            if AP1flag == 1:
                AP1successcounter = AP1successcounter - 1
            else:
                pass
        else:
            cw1rollbackflag, cw2rollbackflag = 1, 1
        #进入下一事件阶段
        time1_old = time1_new
        time2_old = time2_new
    data = 2*AP1successcounter*EP*rate*(10**6)/time1_new
    data_list.append(data)
#excel
work_book = xlwt.Workbook(encoding = 'utf-8')
work_sheet = work_book.add_sheet('sheet')
for i in range(len(data_list)):
    work_sheet.write(i, 0, data_list[i])
work_book.save('experiment_q3.xls')
```

问题四数值求解 Matlab 代码：

改进 Bianchi 模型： main4.m

```
function F = main4(x)
w0 = 16;
wmax = 1024;
r = 6;
rate = 158.4;
m = log(wmax/w0)/log(2);
```

```
w = zeros(1,r+1);
EP = 1500/rate/125000*1000000;
d = 30/rate/125000*1000000 + 13.6 + EP;
Ts = 30/rate/125000*1000000 + 13.6 + EP + 16 + 32 + 43;
Tc = 30/rate/125000*1000000 + 13.6 + EP + 43 + 65;
e = 9;
kk = d/e;
for i = 0:r
    if (i<=m)
        w(i+1) = 2^i*w0;
    elseif (i<=r)
        w(i+1) = 2^m*w0;
    end
end
sum1 = 0;
for i = 0:r
    sum1 = sum1 + x(5)^i * (w(i+1) + 3) / 2;
end
sum2 = 0;
for i = 0:r
    sum2 = sum2 + x(6)^i * (w(i+1) + 3) / 2;
end
s = 0;
for i = 0:r
    s = s + (1-x(5)) * x(5)^i * x(1) * (Ts-2*d)/Ts;
%     s = s + x(5) * x(5)^i * x(1) * (Tc-2*d)/Tc;
    s = s + (w0-kk+w0)/w0/2 * (1-x(5)) * x(5)^i * x(1) * (d)/Ts;
%     s = s + (w(i+1)-kk+w(i+1))/w(i+1)/2 * x(5) * x(5)^i * x(1) * (d)/Tc;
    k = ceil(kk);
    s = s + (w(i+1)-k+1)*(w(i+1)-k)/2/w(i+1) * x(5)^i * x(1);
    l = floor(kk);
    a = 1 - (kk * 100- l*100)/100;
    s = s + a * (w(i+1)-l)/w(i+1) * x(5)^i * x(1);
end
F(1) = x(1) * sum1 - 1;
F(2) = x(2) * sum2 - 1;
F(3) = 1 - s - x(7);
F(4) = x(1) * (1-x(5)^(r+1))/(1-x(5)) - x(3);
F(5) = x(2) * (1-x(6)^(r+1))/(1-x(6)) - x(4);
F(6) = x(4) - x(5);
F(7) = (2*x(3) - x(3)*x(7))-x(6);
```

模型求解：solve4.m

```
w0 = 16;
wmax = 1024;
r = 6;
rate = 158.4;
m = log(wmax/w0)/log(2);
w = zeros(1,r+1);
EP = 1500/rate/125000*1000000;
d = 30/rate/125000*1000000 + 13.6 + EP;
Ts = 30/rate/125000*1000000 + 13.6 + EP + 16 + 32 + 43;
Tc = 30/rate/125000*1000000 + 13.6 + EP + 43 + 65;
e = 9;
kk = d/e;
fun = @main4;
x0 = [0,0,0,0,0,0,0];
x = fsolve(fun,x0)
b001 = x(1);
b002 = x(2);
t1 = x(3);
t2 = x(4);
pact1 = x(5);
pact2 = x(6);
pov = x(7);
S = (t2*(1-pact2) + (2*t1*(1-pact1)))*EP;
t11 = (t2*(1-pact2)+t1*(1-pact1)) * Ts;
t22 = t2 * pact2 * Tc;
t33 = (1 - t2 - t1*(1-pact1)) * e;
S = S/(t11+t22+t33)*rate*1000000
```

问题四仿真求解 python 代码：

```
import random
import xlwt
from math import log
CWmin = 16
CWmax = 1024
r = 32 #最大重传次数
rate = 158.4
Te = 9
EP = 1500/rate/125000*1000000
Ts = 30/rate/125000*1000000 + 13.6 + EP + 16 + 32 + 43
Tc = 30/rate/125000*1000000 + 13.6 + EP + 43 + 65
PMP = 30/rate/125000*1000000 + 13.6 + EP  #PHY hdr + MAC hdr + Payload
SIFS = 16
DIFS = 43
```

```
ACK = 32
ACktimeout = 65
m = log(CWmax/CWmin,2)
#计算数据分布
data_list = []
for i in range(40):
    time1_old, time2_old, time3_old = DIFS, DIFS, DIFS #记录时间
    cw1icounter, cw2icounter, cw3icounter = 0, 0, 0 #AP 数据的发送次数
    cw1jcounter, cw2jcounter, cw3jcounter = 0, 0, 0 #AP 回退翻倍次数
    cw1rollbackflag, cw2rollbackflag, cw3rollbackflag = 1, 1, 1 #判断是否需要产生新的回退
    AP1successcounter, AP2successcounter, AP3successcounter = 0, 0, 0 #AP 传输成功次数计数器
    #离散事件仿真
    for j in range(10**6):
        if cw1icounter <= m:
            cw1jcounter = cw1icounter
        elif m < cw1icounter < r: #超过最大退避阶数
            cw1jcounter = m
        elif cw1icounter >= r: #超过最大重传次数
            cw1icounter, cw1jcounter = 0, 0
        else:
            pass
        if cw2icounter <= m:
            cw2jcounter = cw2icounter
        elif m < cw2icounter < r:
            cw2jcounter = m
        elif cw2icounter >= r:
            cw2icounter, cw2jcounter = 0, 0
        else:
            pass
        if cw3icounter <= m:
            cw3jcounter = cw3icounter
        elif m < cw3icounter < r:
            cw3jcounter = m
        elif cw3icounter >= r:
            cw3icounter, cw3jcounter = 0, 0
        else:
            pass
        if cw1rollbackflag == 1:
            CW1 = random.randint(0,(CWmin*(2**cw1jcounter)-1))
        else:
            pass
```

```
        if cw2rollbackflag == 1:
            CW2 = random.randint(0,(CWmin*(2**cw2jcounter)-1))
        else:
            pass
        if cw3rollbackflag == 1:
            CW3 = random.randint(0,(CWmin*(2**cw3jcounter)-1))
        else:
            pass
        time1_temp = time1_old + CW1*Te
        time2_temp = time2_old + CW2*Te
        time3_temp = time3_old + CW3*Te
        if time1_temp < min(time2_temp,time3_temp): #AP1 最先回退至 0
            AP1successcounter = AP1successcounter + 1
            time1_new = time1_temp + Ts
            time3_new = time3_temp + Ts
            if time1_new < time3_temp:
                time3_new = time3_old #new 回退至 old
                time2_new = time1_new #hold 至 AP1 发送结束
                cw1rollbackflag, cw2rollbackflag, cw3rollbackflag = 1, 0, 0 #AP1 产
生新的回退 AP3 不改变
                cw1icounter = 0 #AP1 重置 AP3 不改变
            else:
                AP3successcounter = AP3successcounter + 1
                time2_new = time3_new #hold 至 AP3 发送结束
                cw1rollbackflag, cw2rollbackflag, cw3rollbackflag = 1, 0, 1 #AP1 和
AP3 产生新的回退
                cw1icounter, cw3icounter = 0, 0 #AP1 和 AP3 成功上传重置 CW
            time2_test = time2_old
            while 1: #计算 AP2 剩余回退间隙
                CW2 = CW2 - 1
                time2_test = time2_test + Te
                if time2_test > time1_temp:
                    break
                else:
                    pass
        elif time3_temp < min(time1_temp,time2_temp): #AP3 最先回退至 0
            AP3successcounter = AP3successcounter + 1
            time1_new = time1_temp + Ts
            time3_new = time3_temp + Ts
            if time3_new < time1_temp:
                time1_new = time1_old
                time2_new = time3_new
                cw1rollbackflag, cw2rollbackflag, cw3rollbackflag = 0, 0, 1
```

```
            cw3icounter = 0
        else:
            AP1successcounter = AP1successcounter + 1
            time2_new = time1_new
            cw1rollbackflag, cw2rollbackflag, cw3rollbackflag = 1, 0, 1
            cw1icounter, cw3icounter = 0, 0
        time2_test = time2_old
        while 1:
            CW2 = CW2 - 1
            time2_test = time2_test + Te
            if time2_test > time3_temp:
                break
            else:
                pass
    elif time2_temp < min(time1_temp,time3_temp): #AP2 最先回退至 0
        AP2successcounter = AP2successcounter + 1
        time2_new = time2_temp + Ts
        time1_new = time2_new
        time3_new = time2_new
        cw1rollbackflag, cw2rollbackflag, cw3rollbackflag = 0, 1, 0
        cw2icounter = 0
        time1_test = time1_old
        while 1:
            CW1 = CW1 - 1
            time1_test = time1_test + Te
            if time1_test > time2_temp:
                break
            else:
                pass
        time3_test = time3_old
        while 1:
            CW3 = CW3 - 1
            time3_test = time3_test + Te
            if time3_test > time2_temp:
                break
            else:
                pass
    elif time1_temp == time2_temp < time3_temp: #AP1 和 AP2 同时回退至 0 冲突
        time1_new = time1_temp + Tc
        time2_new = time2_temp + Tc
        time3_new = time1_new
        cw1rollbackflag, cw2rollbackflag, cw3rollbackflag = 1, 1, 0 #冲突 AP 重
新回退
```

```
            cw1icounter = cw1icounter + 1 #冲突计数器加一
            cw2icounter = cw2icounter + 1
            time3_test = time3_old
            while 1:
                CW3 = CW3 - 1
                time3_test = time3_test + Te
                if time3_test > time1_temp:
                    break
                else:
                    pass
        elif time3_temp == time2_temp < time1_temp: #AP3 和 AP2 同时回退至 0 冲突
            time3_new = time3_temp + Tc
            time2_new = time2_temp + Tc
            time1_new = time3_new
            cw1rollbackflag, cw2rollbackflag, cw3rollbackflag = 0, 1, 1
            cw3icounter = cw3icounter + 1
            cw2icounter = cw2icounter + 1
            time1_test = time1_old
            while 1:
                CW1 = CW1 - 1
                time1_test = time1_test + Te
                if time1_test > time3_temp:
                    break
                else:
                    pass
        elif time1_temp == time3_temp < time2_temp: #AP1 和 AP3 同时回退至 0 均能成功
上传
            AP1successcounter = AP1successcounter + 1
            AP3successcounter = AP3successcounter + 1
            time1_new = time1_temp + Ts
            time3_new = time3_temp + Ts
            time2_new = time1_new
            cw1rollbackflag, cw2rollbackflag, cw3rollbackflag = 1, 0, 1
            cw1icounter, cw3icounter = 0, 0 #AP1 和 AP3 均成功上传
            time2_test = time2_old
            while 1:
                CW2 = CW2 - 1
                time2_test = time2_test + Te
                if time2_test > time1_temp:
                    break
                else:
                    pass
        elif time1_temp == time2_temp == time3_temp: #AP1 和 AP2 和 AP3 同时回退至 0 冲
```

突

```
            time1_new = time1_temp + Tc
            time2_new = time2_temp + Tc
            time3_new = time3_temp + Tc
            cw1rollbackflag, cw2rollbackflag, cw3rollbackflag = 1, 1, 1
            cw1icounter = cw1icounter + 1
            cw2icounter = cw2icounter + 1
            cw3icounter = cw3icounter + 1
        else:
            pass
        #进入下一事件阶段
        time1_old = time1_new
        time2_old = time2_new
        time3_old = time3_new
    data                                                                                    =
(AP1successcounter+AP2successcounter+AP3successcounter)*EP*rate*(10**6)/time1_new
    data_list.append(data)
#excel
work_book = xlwt.Workbook(encoding = 'utf-8')
work_sheet = work_book.add_sheet('sheet')
for i in range(len(data_list)):
    work_sheet.write(i, 0, data_list[i])
work_book.save('experiment_q4.xls')
```