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　北京理工大学

参赛队号　23100070049

队员姓名　1.李政翰

2.刘周杰

3.王一鸣

中国研究生创新实践系列大赛

中国光谷·“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：

无线局域网在日常生活中扮演着重要的作用，而信道接入机制对网络的整体吞吐有重要影响。本文对 Bianchi 模型进行了推广，同时考虑了隐藏终端，暴露终端和自然丢包等非对称复杂网络场景，得到了统一的单节点和网络整体吞吐模型，并通过理论分析比较了不同场景下启用 RTS/CTS 机制对系统吞吐的影响。

为了对理论分析得到的结果进行仿真验证，本文基于 MATLAB 编写了一个离散事件仿真器，实现了对 WLAN 中各节点采用二进制退避算法的分布式信道接入场景的通信性能的仿真。该仿真器模拟了系统节点的退避、等待、传输、冲突和超时重传等状态，可以对系统中每个节点在任意时刻的状态进行可视化展示。仿真器具有较强的可扩展性、可复用性，能够对不局限于本问题的任意复杂网络场景进行事件仿真。

针对**问题一**中 2 BSS 互听且干扰的网络场景，本文首先对 Bianchi 模型进行了推广，考虑了节点的最大重传次数，得到了理论冲突概率p为 10.46%。通过仿真得到的冲突概率为 11.03%，与理论值相差 0.57%，符合实验预期。同时得到了理论吞吐量 67.1744 Mbps，仿真吞吐量 65.4638 Mbps，仿真值与理论值相差 2.55%。此外，我们分析了不同物理层速率、不同数据成功传输概率下 RTS/CTS 机制对理论吞吐的影响。

针对**问题二**中 2 BSS 互听但不干扰的网络场景，将其抽象为暴露终端问题。本文推广得到可以适应暴露终端、隐藏终端、自然丢包等复杂网络场景的统一系统吞吐计算模型。本文将问题二的参数代入该模型，验证了在暴露终端场景下，该模型的正确性。此外，本文说明了在这种场景下，RTS/CTS 机制总会使得系统理论吞吐下降。

针对**问题三**中 2 BSS 不互听、有干扰且系统存在丢包的场景，本文考虑了自然丢包和隐藏终端的影响，定义了脆弱时隙的概念，根据统一系统吞吐计算模型得到各参数下系统理论吞吐量与仿真吞吐量，误差平均在 5%以内。此外，本文通过理论分析证实了在此场景下，RTS/CTS 机制可以有效提升系统理论吞吐。

针对**问题四**中 3 BSS 的网络场景，根据本文提出的统一单节点的吞吐模型，明确了网络中每个节点的互听节点和干扰节点，并对每个节点的碰撞概率和吞吐进行分析，得到了此场景下系统理论吞吐量与仿真吞吐量的误差在 5%左右。

关键词：WLAN；马尔科夫链模型；RTS/CTS；隐藏终端；暴露终端

目 录

1. 问题重述

1.1. 问题背景

随着无线通信技术的高速发展，无线局域网（WLAN, wireless local area network）也即 Wi-Fi 被广泛使用，它提供了低成本、高吞吐和便利的无线通信服务。基本服务集（BSS, basic service set）是 WLAN 的基本组成部分，由站点（STA, station）与无线接入点（AP, access point）组成。常见的 AP 有无线路由器、WiFi 热点等，常见的 STA 有手机、笔记本物联设备等。

随着无线移动终端数量和网络流量的飞速增长，AP 部署日趋高密，然而可用信道有限，使得不同 AP 复用同一信道。AP 使用相同频率发送数据时，存在相互干扰问题，称为同频干扰。针对同频干扰导致的冲突问题，现有的技术通过载波侦听多址接入/退避（CSMA/CA, carrier sense multi-access and collision avoidance）的机制避免冲突，称为分布式协调功能（DCF, distributed coordination function）。

在 CSMA/CA 机制中，能否监听信道和并发传输是否成功是我们要在进行系统建模过程中需要考虑的两个先决条件。前者直接影响节点间的退避时序，后者直接影响数据能否成功传输，从而通过 ACK 间接影响退避时序。

对于前者，是否能监听信道取决于信号能量强度（RSSI，received signal strength indication）与信道可用评估（CCA，clear channel assessment）门限的大小关系。每个节点在发送前都将进行载波侦听，当 RSSI>CCA 门限时，节点将认为信道繁忙，反之节点将认为信道空闲。如果一个节点发送数据时，另一个节点收到来自该节点信号的 RSSI 大于 CCA 门限，则称这两个节点互听。

对于后者，并发传输是否成功取决于信干比（SIR, signal to interference ratio），即信号强度与干扰强度的比值。若 SIR 足够高，则信号能被成功解调，若 SIR 很低，则信号解调失败。

综上所述，本文的研究目标是：在 WLAN 网络中，建立基于 CSMA/CA 机制的多节点通信避免冲突的数学模型，对诸如暴露终端、隐藏终端、自然丢包等多节点场景下的各项通信指标进行系统建模分析。同时基于 MATLAB 对系统建模得到的各项参数进行仿真验证，实现对不同场景下的通信性能指标评估，对实际生活中优化无线网络接入技术起到可行性验证的作用。

1.2. 问题描述

基于上述研究背景，题目提供了 AP 发送数据包的帧结构和物理层信道参数，围绕 WLAN 网络信道接入问题。本文将解决以下四个问题：

问题一：针对提供的系统参数，假设 BSS 间可以互听，且在 AP 并发时数据传输将失败。对该场景下的 2 BSS 系统进行建模，通过数值分析的方法求解，并可以编写仿真器验证模型的精度。

问题二：假设 BSS 间可以互听，且在 AP 并发时数据可以成功传输，即所谓的“暴露终端”问题。对该场景下的 2 BSS 系统进行建模，通过数值分析的方法求解，并可以编写仿真器验证模型的精度。

问题三：在 Bianchi 模型的基础上，考虑非理想无线传输信道，有着 10%的丢包概率。假设 BSS 之间不互听，且在 AP 并发时数据传输将失败，即为隐藏终端问题。对该场景下的 2 BSS 系统进行建模，通过数值分析的方法求解，并可以编写仿真器验证模型的精度。

问题四：考虑 3 BSS 的场景，假设 BSS1 与 BSS3 不互听，BSS2 与两者都互听，且在

AP1 和 AP3 并发时数据可以成功传输。对该场景下的 3 BSS 系统进行建模，通过数值分析的方法求解，并可以编写仿真器验证模型的精度。

2. 模型假设与符号说明

2.1. 模型假设

1. 假设本问题场景下的传播时延充分小[1]，以至于相对于其他时间间隔可以忽略不计。
2. 假设本问题中碰撞概率的取值都是稳态解的固定值，不考虑瞬态碰撞概率的波动。

说明：

关于模型假设 1，之所以可以认为传播时延可以忽略不计，是因为文献[1]指出 WLAN 的室内最大覆盖范围一般是 50-100m，室外的最大覆盖范围是 300m，按照光速约为 $3\times10^8\mathrm{m/s}$计算，则传播时延约为$1\mu\mathrm{s}$，远小于其他时间间隔，因此忽略传播时延是合理的。

关于模型假设 2，严格意义上节点在不同大小的退避窗口下，瞬态的碰撞概率是不相同，从而可能导致碰撞概率会有微弱的波动。但总体而言这个波动较小，且难以估计，故在本问题中只考虑稳态的碰撞概率，并认为节点总是处于稳态。

2.2. 符号说明

本文采用的符号如下表所示：

符号	含义
B	物理层速率
$b_{i,k}$	第i次重传时，剩余时隙为k的状态在二维马尔可夫链的稳态解
$\mathbb{D}_n$	所有与节点n互听且会互干扰的节点（不含n自身）构成的集合
$\mathbb{E}_n$	所有与节点n互听且会不干扰的节点（不含n自身）构成的集合
$E[P]$	数据帧有效载荷的传输时长
$\mathbb{F}_n$	所有与节点n不互听且会干扰的节点（不含n自身）构成的集合
κ,κ_n	节点n与隐蔽站发生冲突的概率
m	二进制指数退避算法的最大退避阶数
$\mathcal{N}$	整个系统内全体节点构成集合
$P_{\mathrm{idle}},P_{\mathrm{idle},n}$	每个时隙T_{slot}节点n所在的互听子信道空闲的概率
$P_{\mathrm{c}},P_{\mathrm{c},n}$	每个时隙节点n所在信道有站点在发送数据，且有站点发送数据冲突的概率
$P_{\mathrm{s}},P_{\mathrm{s},n}$	每个时隙节点n所在信道有站点在发送数据，且没有站点发送数据冲突的概率
p	某个时隙发生碰撞的概率
P_{e}	丢包率
r	最大重传次数
S	标准化系统吞吐量
S_{RTS}	开启 RTS/CTS 模式下的全网吞吐量
T_{TO}	ACK 超时时间
T_{s}	传输成功的时间
T_{c}	传输失败的时间
T_{slot}	退避算法中单个时隙的长度
$T_{\mathrm{s}}^{\mathrm{RTS}}$	采用 RTS/CTS 模式传输成功的时间
$T_{\mathrm{c}}^{\mathrm{RTS}}$	采用 RTS/CTS 模式传输失败的时间

T_{RTS}	传输 RTS 所需时间
T_{CTS}	传输 CTS 所需时间
T_{DATA}	数据包传输时间
T_{hid}	隐蔽站点传输失败所需时间
$T_{\mathrm{in-hid}}$	脆弱时期交叠的期望时长
T_{cov}	覆盖站点传输失败所需时间
$T_{\mathrm{in-cov}}$	脆弱时期交叠的期望时长
$T_{\mathrm{cov}}^{\mathrm{RTS}}$	采用 RTS/CTS 模式覆盖站点传输失败所需时间
$T_{\mathrm{hid}}^{\mathrm{RTS}}$	采用 RTS/CTS 模式隐蔽站点传输失败所需时间
$T_{\mathrm{CTS-TO}}$	接收 CTS 超时时间
V	脆弱时期时长对应的退避时隙数
$W_{\min}$	最小窗口大小
$W_{\max}$	最大窗口大小
W_i	当前窗口大小
X	竞争窗口大于V时的最小退避阶数
τ	节点在某个时隙发送数据帧的概率
$\lambda_{a_1,\cdots,a_N}$	节点$a_1,\cdots,a_N$间是否存在冲突

3. 问题分析与求解思路

3.1. 问题分析

根据问题描述，可以根据 BSS 间能否互听及能否互相干扰，将问题一到问题四表述为图 3-1 至图 3-4 所对应的网络场景，以 AP 为圆心，覆盖范围为半径画圆。假设存在两个基本服务集 $\mathrm{BSS_i}$ 和 $\mathrm{BSS_j}$，如果 $\mathrm{AP_i}$ 和 $\mathrm{AP_j}$ 的距离小于半径，认为 $\mathrm{BSS_i}$ 和 $\mathrm{BSS_j}$ 间能够互听，否则表示不能互听；如果 $\mathrm{AP_i}$ 和 $\mathrm{STA_j}$ 的距离小于半径，则认为 $\mathrm{AP_i}$ 和 $\mathrm{AP_j}$ 并发时不能成功传输数据，否则表示可以成功传输数据。

3.1.1. 问题一

问题一要求针对提供的系统参数，假设 BSS 间可以互听，且 AP 并发时数据不可以成功传输，对 2 BSS 系统的吞吐进行建模。

本问题实际对应的情景是图 3-1 所示的网络场景，即 AP1 和 AP2 可以感知到双方的存在，且当 AP1 和 AP2 同时发送数据时，AP1 对 STA2 和 AP2 对 STA1 的干扰都比较大，数据不能正常传输。

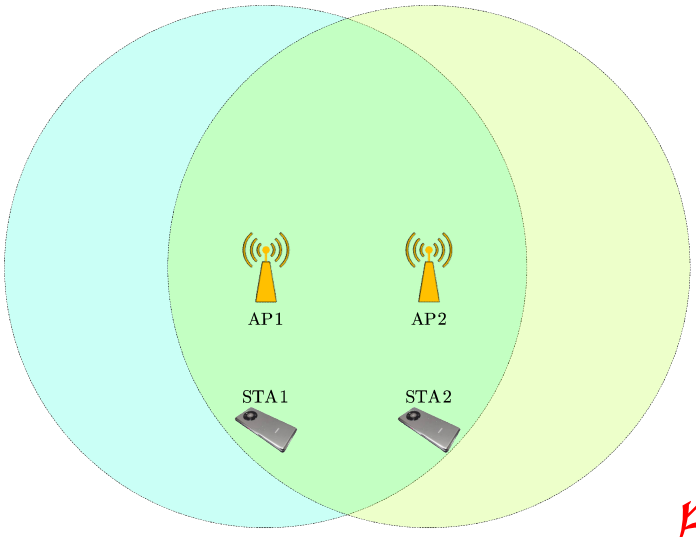

图 3-1 问题一对应的网络场景

3.1.2. 问题二

问题二要求假设 BSS 间可以互听，AP 并发时数据可以成功传输，对 2 BSS 系统的吞吐进行建模分析。

本问题实际对应的情景是图 3-2 所示的网络场景，即 AP1 和 AP2 可以感知到双方的存在，但是当 AP1 和 AP2 同时发送数据时，AP1 对 STA2 和 AP2 对 STA1 的干扰都比较小，数据可以成功传输。这实际对应了网络中暴露终端的概念，也就是当 AP1 在侦听通道时，如果它检测到 AP2 活动，它可能会错误地认为自己不能发送数据，尽管实际上与其通信的节点 STA1 并不会造成干扰，这种现象会降低系统的吞吐量。

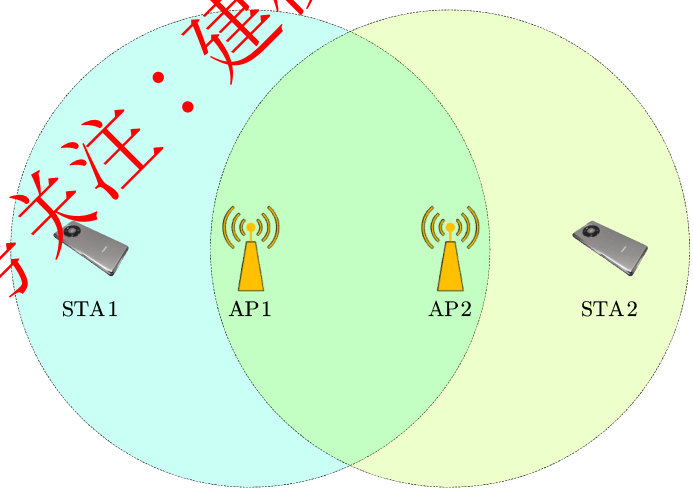

图 3-2 问题二对应的网络场景

3.1.3. 问题三

问题三要求考虑丢包率$P_e = 10\%$的非理想无线传输信道，假设 BSS 之间不互听，且 AP 并发会导致发包失败，对 2 BSS 系统进行建模，求解并评估系统的吞吐。

本问题实际对应的情景是图 3-3 所示的网络场景，即 AP1 和 AP2 不能感知到双方的存在，但是当 AP1 向 STA1 和 AP2 向 STA2 同时发送数据时，数据不能成功传输。这实际对应了网络中隐藏终端的概念，也就是当 AP1 在侦听通道时，检测不到 AP2 的状态，它可

能会错误地认为自己可以向 STA1 发送数据，而实际上如果此时 AP2 也在向 STA2 发送数据，就会因为干扰导致发包失败，这种现象也会降低系统的吞吐量。理论上通过 RTS/CTS 机制，可以在一定程度上减小数据包碰撞带来的系统开销。

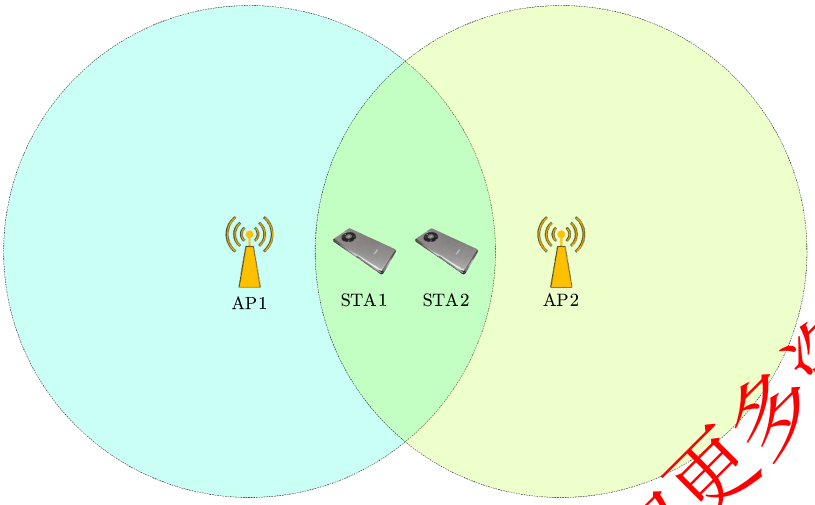

图 3-3 问题三对应的网络场景

3.1.4. 问题四

问题四要求考虑 3 BSS 的场景，假设 BSS1 与 BSS3 不互听，BSS2 与两者都互听，且 AP1 和 AP3 并发时发包成功，对系统进行建模。

本问题实际对应的情景是图 3-4 所示的网络场景，即 AP1 和 AP2，AP2 和 AP3 之间可以感知到双方的存在，AP1 和 AP3 之间不能感知到对方的存在。AP1 向 STA1 发送数据和 AP3 向 STA3 发送数据不会彼此干扰。由于题目中未提及 AP2 与 AP1 或 AP3 时并发时能否成功发包，因此 AP2 向 STA2 的发包过程可能收到 AP1 向 STA1 发送数据或 AP3 向 STA3 发送数据的干扰。具体表现为图 3-4 所示的四种可能情况。

情况一：AP1 与 AP2 之间会互相干扰，AP2 与 AP3 之间会互相干扰。

情况二：AP1 与 AP2 之间会互相干扰，AP2 与 AP3 之间不会互相干扰。

情况三：AP1 与 AP2 之间不会互相干扰，AP2 与 AP3 之间会互相干扰。

情况四：AP1 与 AP2 之间不会互相干扰，AP2 与 AP3 之间不会互相干扰。

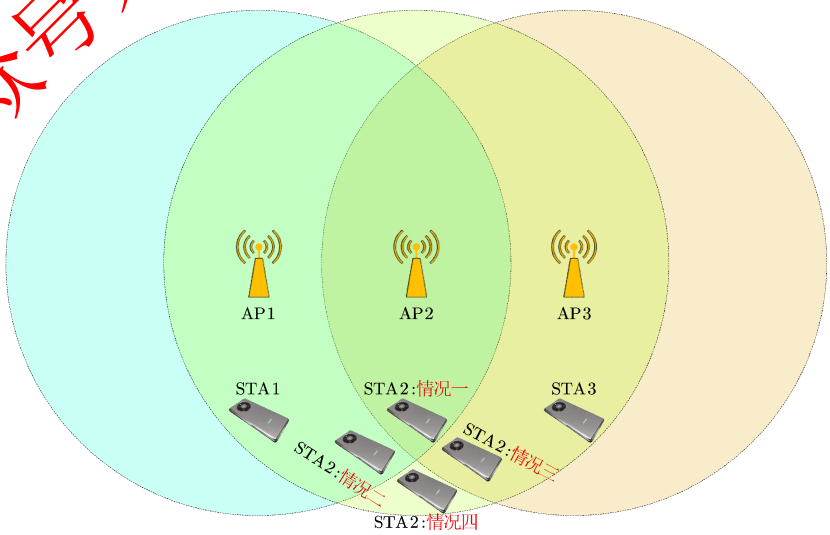

图 3-4 问题四对应的网络场景

3.2. 求解思路

针对上述问题的整体求解思路如下图所示：

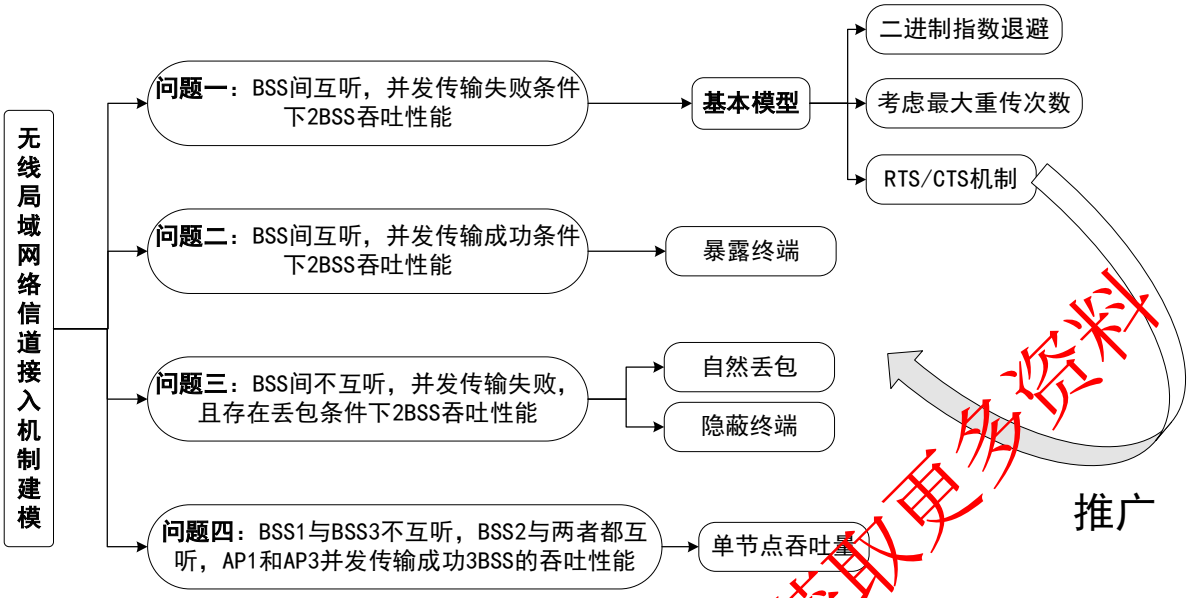

图 3-5 问题求解思路示意图

本文基于 MATLAB 编写了一个离散事件仿真器，具体工作流程如下图 3-6 所示：

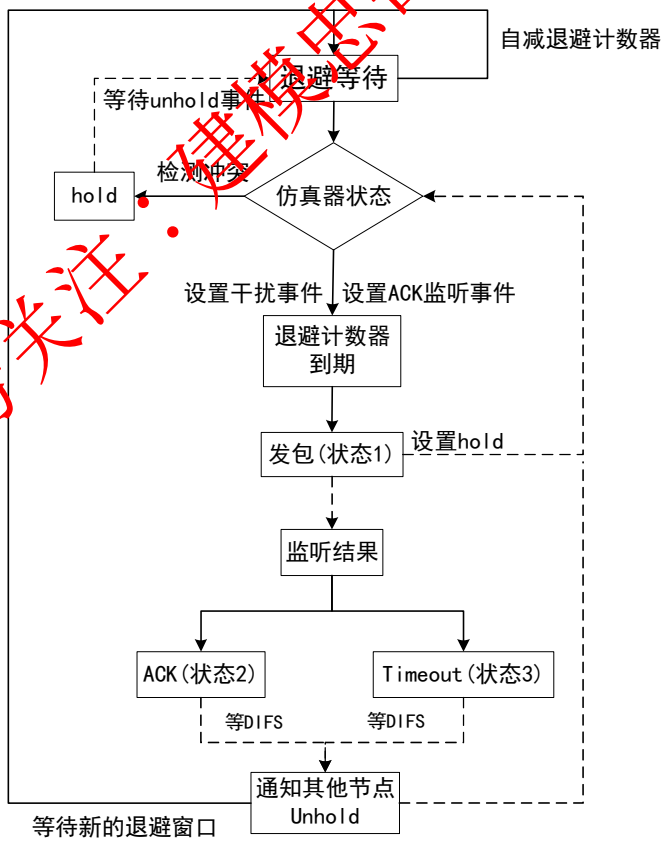

图 3-6 离散事件仿真器工作流程图

该仿真器采用事件驱动模型，将每个节点抽象为了类。每个类中包含的退避、等待、

传输、冲突和超时重传等状态，把各个状态间的切换当成事件，采用事件驱动的方式对整个系统进行仿真模拟，实现了对 WLAN 中各节点事件状态变化的仿真模拟，并且实现了对各种事件的可视化展示。该仿真器的具体工作流程如图 3-6 所示：其中，发包事件对应状态 1，收到 ACK 事件对应状态 2，收到 Timeout 事件对应状态 3，其他情况下对应状态 0。

4. 基本模型

4.1. 随机退避

随机退避算法是一种解决冲突问题的常用方法，适用于多个节点同时访问共享信道资源的情形。在随机退避算法中，当发生冲突时，每个节点会随机选择一个退避时间，然后在该时间后再次尝试发送数据。通过随机选择退避时间，可以避免多个节点同时再次发送数据，从而减少冲突的发生。

其中二进制指数退避算法是一种常见的选择退避时间的方式，该算法的主要思想是按指数倍率逐渐增大竞争窗口 W 的大小，每次退避计数器从$[0, W-1]$中随机选取一个值作为退避时长，以实现冲突避免的效果。竞争窗口的初始值为最小值 $W_{\min}$，之后每次数据传输失败后进行重传时，竞争窗口就会加倍，如果 W 达到了最大值 $W_{\max}$，则保持此值，直到被重置为止。

图 4-1 以三个节点为例说明。图中 W 表示当前阶竞争窗口大小，w 表示随机回退过程时退避计数器从$[0, W-1]$随机选取的初始值。三个节点的 $W_{\min}$ 分别是$8, 16, 32$。开始时，Station c 在发送数据，信道繁忙，数据发送完成后，退避计数器重置。站点 a、b 和 c 持续侦听信道 DIFS 时长，信道被检测为空闲，三者分别开始随机回退，都处于第 0 阶，竞争窗为$[0, W_{\min}-1]$。站点 a 从$[0, 7]$选择了一个随机数 7，需要回退 7 个时隙；站点 b 则从$[0, 15]$选择了随机数 12 回退；站点 c 从$[0, 31]$选择了随机数 16 回退；显然，站点 a 最先回退到 0，抢占到信道，开始一次数据发送，此时，站点 b 和 c 在其回退过程中由于侦听到信道繁忙，随机回退暂停。当站点 a 发送成功后，其竞争窗口重置，信道持续 DIFS 时长空闲后，站点 a 重新从$[0, 7]$选择随机数 5 回退，站点 b 和 c 接着暂停前的回退数继续回退。本次站点 a 和 b 同时回退到 0，同时发送数据，由于冲突导致发送失败，接收节点将不会回复 ACK，站点 a 和 b 在等待 ACK 超时时间后判断数据发送失败，进行重传，将竞争窗翻倍，再次侦听信道 DIFS 时长判断信道空闲后，站点 a 从$[0, 15]$选择随机数 11 回退，站点 b 则从$[0, 31]$选择随机数 9 回退。

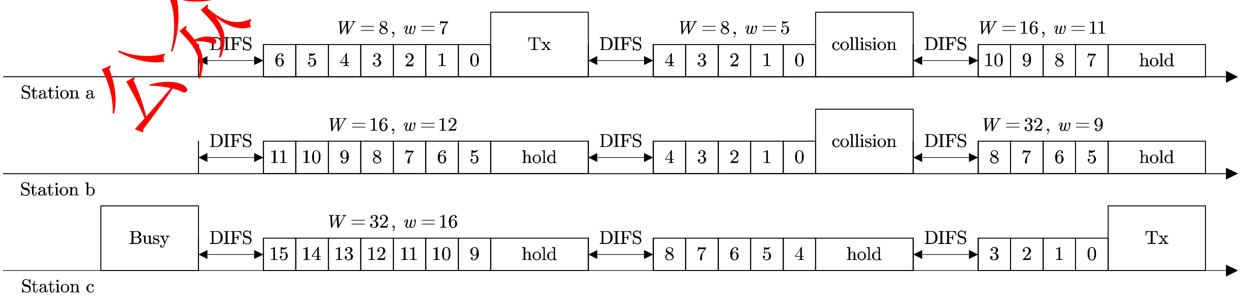

图 4-1 二进制指数退避过程

令 $b(t)$ 和 $s(t)$ 代表 t 时刻一个节点退避随机过程的退避计数和退避阶数，这里的 t 是一个离散的虚拟时隙的开始时刻。用 i 表示一个数据的发送次数，也叫作阶数，r 为最大重传次数，m 是最大退避阶数，则 W 可用下式表示：

$$\begin{cases} W_i = 2^i W_0, & 0 \le i \le m \\ W_i = 2^m W_0, & m \le i \le r \end{cases} \tag{4.1.1}$$

4.2. 简单场景下的冲突概率

在考虑复杂的含隐藏终端、暴露终端、自然丢包、节点间不对称情形的场景之前，我们首先考虑一个简单场景下的冲突概率求解问题。具体而言，在该场景下，我们认为所有节点均互听、会互相干扰，并且自然丢包率为 0。事实上，该简单场景对应了问题一的场景，对该场景的进行建模对于其他场景而言是具有启发意义的。

对于采用二进制指数退避算法的节点，每个时刻t的状态可以由$\{s(t), b(t)\}$描述，其中二维$\{s(t), b(t)\}$随机过程可以用马尔可夫链的形式表示，具体如图 4-2 所示。其中$\lim_{t\to\infty} P\{s(t)=i, b(t)=k\}, i \in [0,m], k \in [0, W_i - 1]$，代表二维马尔可夫过程的稳态解。

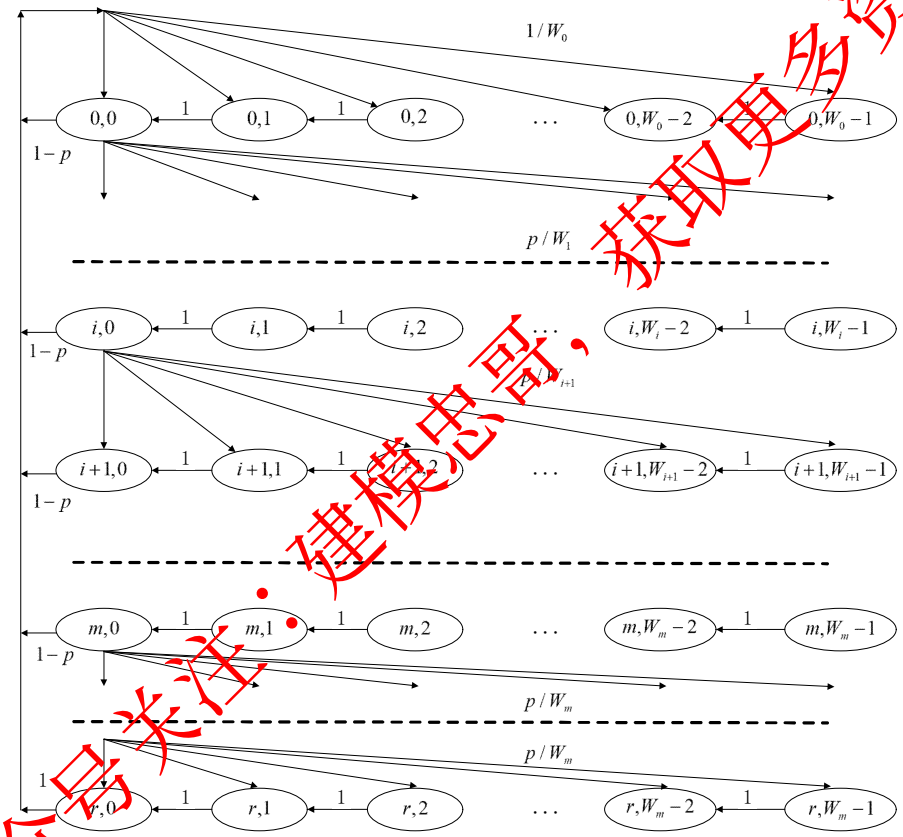

图4-2 DCF的二维马尔可夫链模型

其中，p为某个时隙发生碰撞的概率，马尔可夫链的状态转移方程为：

$$\begin{aligned} &P\{i,k|i,k+1\} = 1, && k \in [0, W_i - 2], i \in [0, r] \\ &P\{0,k|i,0\} = (1-p)/W_0, && k \in [0, W_0 - 1], i \in [0, r-1] \\ &P\{i,k|i-1,0\} = p/W_i, && k \in [0, W_i - 1], i \in [1, r] \\ &P\{0,k|r,0\} = 1/W_0, && k \in [0, W_0 - 1] \end{aligned} \tag{4.2.1}$$

式(4.2.1)中的每个式子分别代表了一定的物理含义。第一个等式代表，未达到退避时长时，退避计数器在每个空闲时隙的开始时刻减 1 的概率是 1。第二个等式代表，未达到重传上限时，当一个数据成功传输后，新到达的数据在$[0,\ W_0 - 1]$中等概率选一个随机数进行回退。第三个等式代表，未达到重传上限时，当一个数据第$i-1$次传输过程发生碰撞，节点进入第i阶回退过程，并在$[0,\ W_i - 1]$中等概率选一个随机数进行回退。最后一个等式

代表，当节点到达最大的传输次数以后，无论成功还是失败，W都会被重置。

该马尔可夫链的任意状态之间可达，是不可约的，且从任意状态到另一状态的步长不存在周期。从任何状态出发，都能到达另一状态，具有常返性。因此该二进制退避过程的非周期不可约马尔可夫链具有稳态解，且所有稳态的概率之和为 1。稳态解可以表示为：

$$b_{i,k} = \lim_{t\to\infty} P\{s(t) = i, b(t) = k\}, i \in [0, m], k \in [0, W_i - 1] \tag{4.2.2}$$

对于一次发送失败的情况，状态$b_{i-1,0}$到状态$b_{i,0}$的步长包括$b_{i-1,0} \to b_{i,0}$，$b_{i-1,0} \to b_{i,1} \to b_{i,0}$，……，$b_{i-1,0} \to b_{i,W_i-1} \to \cdots \to b_{i,1} \to b_{i,0}$，求和可得：

$$b_{i,0} = b_{i-1,0} \cdot \left(p \cdot \frac{1}{W_i} + p \cdot \frac{1}{W_i} \cdot 1 + \cdots + p \cdot \frac{1}{W_i} \cdot 1^{W_i-1}\right) = p \cdot b_{i-1,0}, 0 < i \le r \tag{4.2.3}$$

反复递推迭代可以得出：

$$b_{i,0} = p^i \cdot b_{0,0}, 0 \le i \le r \tag{4.2.4}$$

同理，对于任一状态$b_{i,k}$，若$0 < i < r$，则是从一次发送失败的状态，通过竞争窗口加倍之后转移过来的。若$i = 0$，则是从任一阶发送成功，或达到重传次数限制后转移过来的。因此有：

$$b_{i,k} = \begin{cases} b_{i-1,0} \cdot p \cdot \dfrac{W_i - k}{W_i}, & 0 < i \le r \\ (1-p) \cdot \dfrac{W_i - k}{W_i} \cdot \sum_{j=0}^{r-1} b_{j,0} + \dfrac{W_i - k}{W_i} \cdot b_{r,0}, i = 0 \end{cases} \tag{4.2.5}$$

将式(4.2.4)代入式(4.2.5)中，可得：

$$b_{i,k} = b_{i,0} \cdot \frac{W_i - k}{W_i}, 0 \le i \le r, 0 \le k \le W_i - 1 \tag{4.2.6}$$

根据马尔可夫链的性质，所有稳态的概率之和为 1，因此有：

$$1 = \sum_{k=0}^{W_i-1} \sum_{i=0}^{r} b_{i,k} = \sum_{i=0}^{r} b_{i,0} \sum_{k=0}^{W_i-1} \frac{W_i - k}{W_i} = \sum_{i=0}^{r} b_{i,0} \frac{W_i + 1}{2} \tag{4.2.7}$$

根据式(4.1.1)和式(4.2.7)，可以求得：

$$b_{0,0} = \begin{cases} \dfrac{2(1-p)(1-2p)}{(1-2p)(1-p^{r+1}) + W_0(1-p)(1-(2p)^{r+1})}, & r < m \\ \dfrac{2(1-p)(1-2p)}{W_0(1-(2p)^{m+1})(1-p) + (1-2p)(1-p^{r+1}) + W_0 2^m p^{m+1}(1-p^{r-m})(1-2p)}, & m \le r \end{cases} \tag{4.2.8}$$

考虑到本问题中总有$m \le r$成立，所以可以将式(4.2.8)简化为

$$b_{0,0} = \frac{2(1-p)(1-2p)}{W_0(1-(2p)^{m+1})(1-p) + (1-2p)(1-p^{r+1}) + W_0 2^m p^{m+1}(1-p^{r-m})(1-2p)} \tag{4.2.9}$$

节点随机回退到 0 时发送数据，因此节点在一个时隙发送数据帧的概率为

$$\tau = \sum_{i=0}^{r} b_{i,0} = b_{0,0} \cdot \frac{1 - p^{r+1}}{1 - p} \tag{4.2.10}$$

传输数据发生冲突时，至少有另外一个节点也传输数据，共有 N 个节点，因此条件碰撞概率p可表示为

$$p = 1 - (1 - \tau)^{N-1} \tag{4.2.11}$$

显然地，将式(4.2.9)代入式(4.2.10)，再将式(4.2.10)代入式(4.2.11)，我们就得到了一个

关于 p 的高次多项式方程。这个多项式方程的数值求解过程是容易的，采用牛顿法即可快速求得该多项式方程在$p \in (0,1)$上的实根，将 p 回代入式(4.2.9)、式(4.2.10)也能求出τ。

4.3. 简单场景下的模型吞吐量分析

与 4.2 节类似地，本场景依然考虑的是所有节点均互听、会互相干扰，并且丢包率为 0 的场景。

经过上述分析，如果已知系统中的站点的数量 N，可以计算得到在某个时隙发送数据包时，碰撞概率 p、单个时隙发送数据帧的概率τ的数值解。在此基础上，我们可以通过以下分析得到系统总吞吐量 S 的数值解。

假设 P_{idle} 代表在某个时隙，互听信道中没有站点在发送数据的概率，P_{s} 代表该时隙有站点在发送数据，且没有站点发送数据冲突的概率，P_{c} 代表该时隙有站点在发送数据，且至少有一个站点发送数据冲突的概率，则P_{idle}、P_{s}、P_{c}可以表示为：

$$P_{\text{idle}} = (1-\tau)^N \tag{4.3.1}$$

$$P_{\text{s}} = N\tau(1-\tau)^{N-1} \tag{4.3.2}$$

$$P_{\text{c}} = 1 - P_{\text{idle}} - P_{\text{s}} \tag{4.3.3}$$

让我们简单阐述一下每个量的含义：

- 由于每个节点在某个时隙不发数据的概率是$1-\tau$，信道空闲概率自然是所有节点都不发数据，即为连乘：$P_{\text{idle}} = (1-\tau)^N$
- 由于所有节点共处一个信道且互相干扰，那么信道在该时隙有节点传输成功的概率等价于恰好有且仅有一个节点发送的概率，即$P_{\text{s}} = N\tau(1-\tau)^{N-1}$。
- 注意到概率归一性条件$P_{\text{idle}} + P_{\text{c}} + P_{\text{s}} = 1$，则有$P_{\text{c}} = 1 - P_{\text{idle}} - P_{\text{s}}$.

下图 4-1 反应了某一时刻的信道状态转移图，在某一抽样时刻，信道有P_{idle}的概率空闲，有P_{s}的概率均发送成功的状态，有P_{c}的概率存在至少一个节点发生冲突导致发送失败。

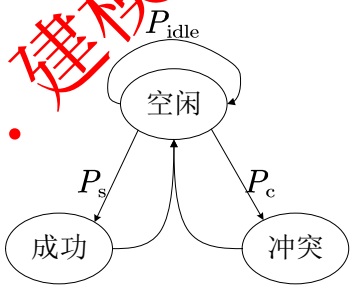

图 4-1 某一时刻的信道状态转移图

吞吐量可以由下公式确定：

$$S = \frac{E\left[\text{一个时隙内传输的有效载荷发送时长}\right]}{E\left[\text{一个时隙长度}\right]} \times \text{物理层速率} \tag{4.3.4}$$

注意到每个节点的时隙期望时长为$T_{\text{slot}} \cdot P_{\text{idle}} + P_{\text{s}} \cdot T_{\text{s}} + P_{\text{c}} \cdot T_{\text{c}}$，根据以上定义，我们可以确定系统的总吞吐量 S，表示为：

$$S = \frac{P_{\text{s}} \cdot E[P]}{T_{\text{slot}} \cdot P_{\text{idle}} + P_{\text{s}} \cdot T_{\text{s}} + P_{\text{c}} \cdot T_{\text{c}}} \tag{4.3.5}$$

其中，$E[P]$代表平均有效载荷的长度，T_{s} 代表数据成功传输的平均时间，T_{c} 代表数据传输时存在冲突的平均时间，T_{slot}代表退避算法中单个时隙的长度。T_{s} 和T_{c} 的值可以由下面的公式确定：

$$\begin{cases} T_s = T_{\mathrm{PHY}} + T_{\mathrm{MAC}} + T_{\mathrm{PLD}} + T_{\mathrm{SIFS}} + T_{\mathrm{ACK}} + T_{\mathrm{DIFS}} \\ T_c = T_{\mathrm{PHY}} + T_{\mathrm{MAC}} + T_{\mathrm{PLD}} + T_{\mathrm{TO}} + T_{\mathrm{DIFS}}, \end{cases} \tag{4.3.6}$$

如果在该场景下采用 RTS/CTS 协议，则会得到不同的结果，但推导方式是类似的。RTS/CTS 是一种用于无线通信的协议机制，其核心思想是用 RTS 短包的碰撞来避免数据包长包的碰撞。在发送方想要发送数据包时，首先发送包含有数据帧发送时间信息的 RTS 帧，如果没有发生冲突，接收方会返回 CTS 帧。发送方成功收到 CTS 帧后，开始数据帧的传输。将传输成功和传输失败的时间记为T_s^{RTS}和T_c^{RTS}，其计算方式如下式所示：

$$\begin{cases} T_s^{\mathrm{RTS}} = T_{\mathrm{RTS}} + T_{\mathrm{SIFS}} + T_{\mathrm{CTS}} + T_{\mathrm{SIFS}} + T_{\mathrm{PHY}} + T_{\mathrm{MAC}} + T_{\mathrm{PLD}} + T_{\mathrm{SIFS}} + T_{\mathrm{ACK}} + T_{\mathrm{DIFS}} \\ T_c^{\mathrm{RTS}} = T_{\mathrm{RTS}} + T_{\mathrm{TO}} + T_{\mathrm{DIFS}} \end{cases} \tag{4.3.7}$$

比较式(4.3.6)和式(4.3.7)可知，在数据包传输成功时，T_s^{RTS}相较于T_s增加了 RTS 和 CTS 帧的开销。当数据包传输失败时，T_c^{RTS}相较于T_c将发送整个数据包的开销减少为发送 RTS 的开销。因此在发送的数据包较大，或发送数据时碰撞概率较高时，可以考虑启用 RTS/CTS 机制。或者在系统存在隐藏终端问题时，采用 RTS/CTS 机制也能有效降低碰撞概率。

用T_s^{RTS}和T_c^{RTS}替换T_s和T_c，即可得到使用 RTS/CTS 机制的吞吐量计算方法：

$$S = \frac{P_s \cdot E[P]}{T_{\mathrm{slot}} \cdot P_{\mathrm{idle}} + P_s \cdot T_s^{\mathrm{RTS}} + P_c \cdot T_c^{\mathrm{RTS}}} \tag{4.3.8}$$

4.4. 对隐藏终端场景下的分析

以上算法考虑的是发送节点之间可以互听的场景，然而，如果存在隐藏节点和自然丢包的问题，碰撞概率p的计算模型会更加复杂。为了简化分析过程，本节内容假设所有节点的冲突概率 p 均相等、各节点拓扑存在对称性，并且不考虑暴露终端问题问题。本文在 4.5 节将考虑包含所有上述情形的场景，提出一个综合考虑了所有情形的大一统公式。

当存在隐藏节点问题时，考虑将节点类型分成覆盖节点和隐藏节点。其中，覆盖节点之间可以互听，分析方式与前文相同。隐藏节点之间不能互听，但是当隐藏节点同时发送数据时会出现冲突，进而影响系统的吞吐。下面对隐藏节点进行建模分析。

首先定义脆弱时期的概念，其含义是由于隐藏节点的存在可能导致数据冲突的时间段。在这段时间内，如果节点发送数据，就会产生冲突。我们定义 V 代表脆弱时期时长对应的退避时隙数，X是竞争窗口大于V时的最小退避阶数。以数据包为例，$V_{\mathrm{PLD}} = T_{\mathrm{PLD}}/T_{\mathrm{slot}}$，其中$T_{\mathrm{PLD}} = T_{\mathrm{PHY}} + (L_{\mathrm{MAC}} + L_{\mathrm{PLD}})/B$。

此时由于隐藏节点无法侦听到本节点，所以在$b_{i,0}, b_{i,1}, \cdots, b_{i,V}$时发送都会导致冲突的发送，因此定义与隐藏节点发生冲突的概率为$\kappa = \sum_{i=0}^{m}\sum_{k=0}^{V} b_{i,k}$，代入式(4.2.6)和式(4.2.4)有：

$$\kappa = \begin{cases} \left[(V+1)\cdot\dfrac{1-p^{m+1}}{1-p} - \dfrac{V(V+1)}{2W_0}\cdot\dfrac{1-\left(\frac{p}{2}\right)^{m+1}}{1-\left(\frac{p}{2}\right)}\right]\cdot b_{0,0}, & V \le W_0 \\ \left[\begin{array}{l} \dfrac{1-p^x}{2(1-p)} + (V+1)\cdot\dfrac{p^x - p^{m+1}}{1-p} \\ + \dfrac{W_0}{2}\cdot\dfrac{1-(2p)^x}{1-2p} - \dfrac{V(V+1)}{2W_0}\cdot\dfrac{\left(\frac{p}{2}\right)^x - \left(\frac{p}{2}\right)^{m+1}}{1-\frac{p}{2}} \end{array}\right]\cdot b_{0,0}, & \begin{array}{l} W_{X-1} < V \le W_X \\ 1 \le X \le m \end{array} \\ 1, & V > W_m \end{cases} \tag{4.4.1}$$

注意到本题目中$T_{\mathrm{PHY}} = 13.6\mu s$，$L_{\mathrm{MAC}} + L_{\mathrm{PLD}} = 1530\mathrm{Byte}$，并且对于每一个问题恒有$B \ge 158.4\mathrm{Mbps}$成立，所以本题目中$T_{\mathrm{DATA}} = T_{\mathrm{PHY}} + (L_{\mathrm{MAC}} + L_{\mathrm{PLD}})/B \le 90.873\mu s$。而本题

目中每一个问题都有$W_0 \geq 16$成立，自然有$V = T_{\mathrm{DATA}}/T_{\mathrm{slot}} < 16 \leq W_0$，所以可以简化为：

$$\kappa = \left[(V+1) \cdot \frac{1-p^{m+1}}{1-p} - \frac{V(V+1)}{2W_0} \cdot \frac{1-\left(\frac{p}{2}\right)^{m+1}}{1-\left(\frac{p}{2}\right)} \right] \cdot b_{0,0} \tag{4.4.2}$$

不难注意到，式(4.4.2)代入$V = 0$即可得到式(4.2.10)的形式，故可以认为τ是κ在$V = 0$时的特例。

定义与本节点互听的站点数量为N_{cov}，隐藏节点的数量为N_{hid}，丢包率为P_{e}，可以计算出冲突概率p为：

$$p = 1 - (1-P_{\mathrm{e}})(1-\tau)^{N_{\mathrm{cov}}-1}(1-\kappa)^{N_{\mathrm{hid}}} \tag{4.4.3}$$

进一步地，我们还可以对存在隐藏节点的系统吞吐进行建模。根据式(4.3.1)可以知道：在某个时隙，互听信道中没有站点在发送数据的概率$P_{\mathrm{idle}} = (1-\tau)^{N_{\mathrm{cov}}}$。

分析可知，节点发送数据时出现碰撞，或者存在隐藏节点问题，都会使节点传输数据失败。可以求得节点成功传输数据的概率为：

$$P_{\mathrm{s}} = N_{\mathrm{cov}} \cdot \tau \cdot (1-p) \tag{4.4.4}$$

由归一化条件，自然有$P_{\mathrm{c}} = 1 - P_{\mathrm{idle}} - P_{\mathrm{s}}$。设系统总节点数为$N_{\mathrm{all}}$，进而得到系统的总吞吐量为：

$$S = \frac{N_{\mathrm{all}} \cdot \tau \cdot (1-p) \cdot E[P]}{T_{\mathrm{slot}} \cdot P_{\mathrm{idle}} + P_{\mathrm{s}} \cdot T_{\mathrm{s}} + P_{\mathrm{c}} \cdot T_{\mathrm{c}}} \tag{4.4.5}$$

其中，T_{s}和T_{c}可以表示为

$$\begin{cases} T_{\mathrm{s}} = T_{\mathrm{PHY}} + T_{\mathrm{MAC}} + T_{\mathrm{PLD}} + T_{\mathrm{SIFS}} + T_{\mathrm{ACK}} + T_{\mathrm{DIFS}} \\ T_{\mathrm{c}} = \dfrac{N_{\mathrm{cov}}}{N_{\mathrm{cov}}+N_{\mathrm{hid}}} \cdot T_{\mathrm{cov}} + \dfrac{N_{\mathrm{hid}}}{N_{\mathrm{cov}}+N_{\mathrm{hid}}} \cdot T_{\mathrm{hid}} \\ T_{\mathrm{hid}} = T_{\mathrm{PHY}} + T_{\mathrm{MAC}} + T_{\mathrm{PLD}} + T_{\mathrm{TO}} + T_{\mathrm{DIFS}} + T_{\mathrm{in-h}} \\ T_{\mathrm{in-hid}} = \dfrac{1}{2}(T_{\mathrm{PHY}} + T_{\mathrm{MAC}} + T_{\mathrm{PLD}}) \\ T_{\mathrm{cov}} = T_{\mathrm{PHY}} + T_{\mathrm{MAC}} + T_{\mathrm{PLD}} + T_{\mathrm{TO}} + T_{\mathrm{DIFS}} + T_{\mathrm{in-c}} \\ T_{\mathrm{in-cov}} = \dfrac{1}{2} \cdot T_{\mathrm{slot}} \end{cases} \tag{4.4.6}$$

此处$T_{\mathrm{in-hid}}$、$T_{\mathrm{in-}}$项代表脆弱时期交叠的期望时长，对于隐藏终端问题而言，通常两个隐藏节点的时钟没有对齐，可以视为$[0, V]$上的均匀分布，则为$V/2$。如果时钟对齐，则可视为 0。

在此基础上，我们考虑使用 RTS/CTS 的场景，用$T_{\mathrm{s}}^{\mathrm{RTS}}$和$T_{\mathrm{c}}^{\mathrm{RTS}}$代替$T_{\mathrm{s}}$和$T_{\mathrm{c}}$，$T_{\mathrm{s}}^{\mathrm{RTS}}$和$T_{\mathrm{c}}^{\mathrm{RTS}}$可以表示为，$T_{\mathrm{CTS-T}}$ 代表 CTS 发生超时所用的时长：

$$\begin{cases} T_{\mathrm{s}}^{\mathrm{RTS}} = T_{\mathrm{RTS}} + T_{\mathrm{SIFS}} + T_{\mathrm{CTS}} + T_{\mathrm{SIFS}} + T_{\mathrm{PHY}} + T_{\mathrm{MAC}} + T_{\mathrm{PLD}} + T_{\mathrm{SIFS}} + T_{\mathrm{ACK}} + T_{\mathrm{DIFS}} \\ T_{\mathrm{c}}^{\mathrm{RTS}} = \dfrac{N_{\mathrm{cov}}}{N_{\mathrm{cov}}+N_{\mathrm{hid}}} \cdot T_{\mathrm{cov}}^{\mathrm{RTS}} + \dfrac{N_{\mathrm{hid}}}{N_{\mathrm{cov}}+N_{\mathrm{hid}}} \cdot T_{\mathrm{hid}}^{\mathrm{RTS}} \\ T_{\mathrm{cov}}^{\mathrm{RTS}} = \dfrac{1}{2} \cdot T_{\mathrm{slot}} + T_{\mathrm{RTS}} + T_{\mathrm{CTS-}} \\ T_{\mathrm{hid}}^{\mathrm{RTS}} = \dfrac{3}{2} \cdot T_{\mathrm{RTS}} + T_{\mathrm{CTS-T}} \end{cases} \tag{4.4.7}$$

进而得到系统的总吞吐量为：

$$S = \frac{N_{\mathrm{all}} \cdot \tau \cdot (1-p) \cdot E[P]}{T_{\mathrm{slot}} \cdot P_{\mathrm{idle}} + P_{\mathrm{s}} \cdot T_{\mathrm{s}}^{\mathrm{RTS}} + P_{\mathrm{c}} \cdot T_{\mathrm{c}}^{\mathrm{RTS}}} \tag{4.4.8}$$

4.5. 对所有情形都适用的统一公式

在 4.2、4.3 节，我们考虑了一种最简单的情形，这种情形不考虑丢包、暴露站点、隐藏站点，从而引入了马尔科夫链的思想对问题进行求解。进一步地，我们在 4.4.节考虑了隐藏站点和可能自然丢包的情形，得到了一个更一般的公式，但此时并没有考虑暴露站点的情形。在本节，我们将综合前面的所有工作，并引入对暴露站点情形的分析，得到一个大一统但复杂的公式。虽然公式形式看似复杂，但在代入具体数值后并不复杂，并且本公式十分利于编程实现，可以推广到任意场景下。

首先，我们定义三类节点构成的集合：

1. 记通信环境中全体与节点n互听且会互干扰的节点（不含n自身）构成的集合为$\mathbb{D}_n$。
   具体而言，$\mathbb{D}_n = \{k|\mathrm{RSSI}_{k\to n} > \mathrm{CCA}_{\mathrm{threshold}}, \mathrm{SIR}_{k\to n} < \mathrm{SIR}_{\mathrm{threshold}}, k \neq n\}$。
2. 记通信环境中全体与节点n互听且会不干扰的节点（不含n自身）构成的集合为$\mathbb{E}_n$，这也就是所谓的暴露站点。
   具体而言，$\mathbb{E}_n = \{k|\mathrm{RSSI}_{k\to n} > \mathrm{CCA}_{\mathrm{threshold}}, \mathrm{SIR}_{k\to n} > \mathrm{SIR}_{\mathrm{threshold}}, k \neq n\}$。
3. 记通信环境中全体与节点n不互听且会干扰的节点（不含n自身）构成的集合为$\mathbb{F}_n$，这也就是所谓的隐藏站点。
   具体而言，$\mathbb{F}_n = \{k|\mathrm{RSSI}_{k\to n} < \mathrm{CCA}_{\mathrm{threshold}}, \mathrm{SIR}_{k\to n} < \mathrm{SIR}_{\mathrm{threshold}}, k \neq n\}$。

对于任意节点n，每次发包的冲突概率为p_n，单个时隙发送数据帧的概率为τ_n，与(4.2.10)类似地，仍然有：

$$\tau_n = b_{0,0,n} \cdot \frac{1-(p_n)^{r+1}}{1-p_n} \tag{4.5.1}$$

$b_{0,0,n}$代表节点 n 的$b_{0,0}$，由式(4.2.9)可得

$$b_{0,0,n} = \frac{2(1-p_n)(1-2p_n)}{W_0(1-(2p_n)^{m+1})(1-p_n) + (1-2p_n)(1-(p_n)^{r+1}) + W_0 2^m (p_n)^{m+1}(1-(p_n)^{r-m})(1-2p_n)} \tag{4.5.2}$$

而冲突概率p_n等于 1 减去其他干扰节点$\mathbb{D}_n \cup \mathbb{F}_n$在此时恰好不发包的概率乘以不丢包率，即：

$$p_n = 1-(1-P_e)\prod_{k\in\mathbb{D}_n}(1-\tau_k)\prod_{k\in\mathbb{F}_n}(1-\kappa_k) \tag{4.5.3}$$

其中κ_k是节点k与隐藏站点发生冲突的概率，与式(4.4.2)类似地，有

$$\kappa_k = \left[(V+1)\cdot\frac{1-(p_k)^{m+1}}{1-p_k} - \frac{V(V+1)}{2W_0}\cdot\frac{1-\left(\frac{p_k}{2}\right)^{m+1}}{1-\left(\frac{p_k}{2}\right)}\right]\cdot b_{0,0,k} \tag{4.5.4}$$

其中$V = T_{\mathrm{DATA}}/T_{\mathrm{slot}}$，$T_{\mathrm{DATA}} = T_{\mathrm{PHY}} + (L_{\mathrm{MAC}} + L_{\mathrm{PLD}})/B$。

假设全系统的节点集合为$\mathcal{N}$，显然地，由于τ_k和κ_k均为关于p_k的有理多项式，故式(4.5.3)是关于$p_k, \forall k \in \mathcal{N}$的多项式方程。故对每个节点都列出式(4.5.3)，就得到了一个N元的多项式方程组，联立求解即可得到每个节点的冲突概率p_n。

与式(4.3.1)类似地，可以自然推导出：

$$P_{\mathrm{idle},n} = (1-\tau_n)\prod_{k\in\mathbb{D}_n}(1-\tau_k)\prod_{k\in\mathbb{E}_n}(1-\tau_k) \tag{4.5.5}$$

记节点n所在的互听信道有站点在发送数据，且没有一个节点发送数据有冲突的概率$P_{\mathrm{s},n}$，由于本问题考虑了暴露站点问题，故它与式(4.4.4)看起来略有区别。我们用$\lambda_{a_1,\cdots,a_N}$表

征节点$a_1,\cdots,a_N$间是否存在冲突：

$$\lambda_{a_1,\cdots,a_N}=\begin{cases}1, a_1,\cdots,a_N\text{发包时均互不冲突}\\0, a_1,\cdots,a_N\text{发包时至少有一对节点冲突}\end{cases} \tag{4.5.6}$$

将$a_1,\cdots,a_N$发包时均互不冲突用严格的数学语言表述为$\forall i\neq j, \mathrm{SIR}_{i\to j}>\mathrm{SIR}_{\mathrm{threshold}}$。

此时设全体与 n 互听节点构成的集合$\{n\}\cup\mathbb{D}_n\cup\mathbb{E}_n$的元素数量为$N_n$，则可以得到：

$$P_{\mathrm{s},n}=1-\sum_{0\le\mu_1,\cdots,\mu_{N_n}\le 1}\left((-1)^{\mu_1+\cdots+\mu_{N_n}}\cdot\lambda_{\{k|\mu_k=1\}}\cdot\prod_{i\in[1,N_n]}(\tau_i)^{\mu_i}(1-p_i)^{\mu_i}\right) \tag{4.5.7}$$

式(4.5.7)看似复杂，但其思想是简单的，事实上这就是对N元多项式展开公式的简单变形。我们用布尔值μ_k表示每个节点发包与否，则这组事件的发生概率为$(\tau_1)^{\mu_1}(\tau_2)^{\mu_2}\cdots(\tau_{N_n})^{\mu_{N_n}}$，此时每个节点$k$的自然发包成功概率为$(1-p_k)^{\mu_k}$。但如果在集合$\{k|\mu_k=1\}$中存在一对节点互相冲突，那么发包成功概率一定为 0，这就是$\lambda_{\{k|\mu_k=1\}}$的作用。

此时该子信道的时隙期望时长为$T_{\mathrm{slot}}\cdot P_{\mathrm{idle},n}+P_{\mathrm{s},n}\cdot T_{\mathrm{s}}+P_{\mathrm{c},n}\cdot T_{\mathrm{c}}$，其中$P_{\mathrm{c},n}=1-P_{\mathrm{idle},n}-P_{\mathrm{s},n}$，$T_{\mathrm{s}}$和$T_{\mathrm{c}}$见式(4.4.6)，则节点$n$的吞吐量为

$$S_n=\frac{\tau_n\cdot(1-p_n)\cdot E[P]}{T_{\mathrm{slot}}\cdot P_{\mathrm{idle},n}+P_{\mathrm{s},n}\cdot T_{\mathrm{s}}+P_{\mathrm{c},n}\cdot T_{\mathrm{c}}} \tag{4.5.8}$$

假设全网的节点集合为$\mathcal{N}$，则对全体节点的吞吐量求和即为系统总吞吐量：

$$S=\sum_{n\in\mathcal{N}}S_n=\sum_{n\in\mathcal{N}}\frac{\tau_n\cdot(1-p_n)\cdot E[P]}{T_{\mathrm{slot}}\cdot P_{\mathrm{idle},n}+P_{\mathrm{s},n}\cdot T_{\mathrm{s}}+P_{\mathrm{c},n}\cdot T_{\mathrm{c}}} \tag{4.5.9}$$

5. 问题一的分析与求解

5.1. 问题分析

问题一考虑 BSS 间可以互听，且在 AP 并发时数据传输将失败的场景。对该场景下的 2 BSS 系统进行建模。

首先明确此问题中的 BSS 间可以互听，因此不存在隐蔽站问题，且在 AP 并发时数据传输将失败，因此也不存在暴露站问题。其对应于 4.2 和 4.3 节中涉及到的简单场景下的冲突概率和吞吐量建模模型。

与原始文献[1]相比，此问题考虑了最大重传次数，即当节点的重传次数达到最大值r时，如果再次竞争时隙，不论竞争是否成功，都将退回至最小退避阶数，即

$$P\{0,k|r,0\}=\frac{1}{W_0},\qquad k\in[0,W_0-1] \tag{5.1.1}$$

由于两种模型在退避阶数达到最大值采取的不同策略，这将导致式(4.2.9)和式(4.2.10)的结论并不相同。具体而言，由于 Bianchi 模型在最大退避阶数m状态下竞争失败会一直维持该状态，此时 Bianchi 的马尔科夫链相当于一个无限长的马尔可夫序列，因此在求解马尔科夫链稳态解过程中对应了无穷级数求和问题。而本文模型在达到最大重传次数r时，不论如何都会重置退避阶数，这对应了有限长数列求和问题。如果我们考虑$r\to+\infty$的场景，此时我们的模型将退化为 Bianchi 的经典模型，事实上，将$r\to+\infty$代入(4.2.9)和式(4.2.10)，则两式的结论将与 Bianchi 的经典模型一致。

同时，在 Bianchi 的模型中，需要考虑传播时延，而根据模型假设 1，在本题中则忽略传播时延。事实上，忽略传播时延的确是合理的，因为根据文献[1]的数据，一般 WLAN 的

最大覆盖范围约为 300m，按照光速$3\times10^8\mathrm{m/s}$计算，一个电磁波的传播时延也只有$1\mu s$，远小于本问题中的其他参数。

根据以上分析和 4.2 和 4.3 节中提到的理论模型，由式(4.2.9)、式(4.2.10)和式(4.2.11)可以得到某时刻节点发包的冲突概率p。根据式(4.3.5)，可以得到系统吞吐量S，根据式(4.3.8)，可以得到开启 RTS/CTS 模式下的系统吞吐量S_{RTS}。

5.2. 模型建立

问题一模型建立在 4.1、4.2 和 4.3 节中完成，对文献[2]的无限长马尔可夫序列进行了推广，得到了最大重传次数约束条件下的系统归一化的吞吐量 S 及开启 RTS/CTS 模式下的系统吞吐量S_{RTS}。

由于在 4.5 节处我们推导出了一个更一般的模型，该模型可以用于解决所有场景下的问题。此处我们将基于这个一般化的模型代入本问题的条件进行推导，最终将推导出与 4.2、4.3 节一致的结论，从而展示这个一般化模型的用途与正确性。

对于问题一而言，全网只有两个节点，分别记为 1、2。它们两者互听且互干扰，因此有$\mathbb{D}_1=\{2\},\mathbb{D}_2=\{1\},\mathbb{E}_1=\emptyset,\mathbb{E}_2=\emptyset,\mathbb{F}_1=\emptyset,\mathbb{F}_2=\emptyset$，本问题中的丢包率$P_{\mathrm{e}}=0$。

对于每一个节点，由式(4.5.3)，代入$P_{\mathrm{e}}=0,\mathbb{D}_1=\{2\},\mathbb{D}_2=\{1\},\mathbb{F}_1=\emptyset,\mathbb{F}_2=\emptyset$可以得到：

$$\begin{cases}p_1=1-\left(1-b_{0,0,2}\cdot\dfrac{1-(p_2)^{r+1}}{1-p_2}\right)\\ p_2=1-\left(1-b_{0,0,1}\cdot\dfrac{1-(p_1)^{r+1}}{1-p_1}\right)\end{cases}\tag{5.2.1}$$

其中$b_{0,0,k}$的表达式见式(4.5.2)，$b_{0,0,k}$是关于p_k的有理多项式。

显然p_1和p_2的表达式具有交换对称性，并且节点 1、2 的拓扑也具有交换对称性，可以得到$p_1=p_2$。因此我们不必直接求解这个二元多项式方程组，直接代入$p_1=p_2$就退化为了与式(4.2.11)一致的一元多项式方程组了，这也验证了我们的推广模型与简单模型的一致性。

对于$P_{\mathrm{idle},n}$，由式(4.5.5)，代入$\mathbb{D}_1=\{2\},\mathbb{D}_2=\{1\},\mathbb{E}_1=\emptyset,\mathbb{E}_2=\emptyset$，可见两者的$P_{\mathrm{idle},n}$均为$(1-\tau_1)(1-\tau_2)$，代入$\tau=\tau_1=\tau_2$可知与式(4.3.1)一致，即$P_{\mathrm{idle}}=P_{\mathrm{idle},1}=P_{\mathrm{idle},2}$。

对于$P_{\mathrm{s},n}$，由式(4.5.7)，注意到该系统中两节点均互相干扰，故$\lambda_{1,2}=0$，此时$P_{\mathrm{s},n}$退化为$P_{\mathrm{s},n}=\sum_{i\in\{n\}\cup\mathbb{D}_n\cup\mathbb{E}_n}\left(\tau_i\prod_{j\neq i}(1-p_j)\right)$，代入$P_{\mathrm{e}}=0,\mathbb{D}_1=\{2\},\mathbb{D}_2=\{1\},\mathbb{E}_1=\emptyset,\mathbb{E}_2=\emptyset$，有：

$$\begin{cases}P_{\mathrm{s},1}=\tau_1\cdot(1-\tau_2)+\tau_2\cdot(1-\tau_1)\\ P_{\mathrm{s},2}=\tau_1\cdot(1-\tau_2)+\tau_2\cdot(1-\tau_1)\end{cases}\tag{5.2.2}$$

代入$\tau=\tau_1=\tau_2$可知与式(4.3.2)一致，即$P_{\mathrm{s}}=P_{\mathrm{s},1}=P_{\mathrm{s},2}$。

此时对于每个节点而言，由式(4.5.8)，可得单个节点的吞吐量为：

$$\begin{cases}S_1=\dfrac{\tau_1\cdot(1-\tau_2)\cdot E[P]}{T_{\mathrm{slot}}\cdot P_{\mathrm{idle},1}+P_{\mathrm{s},1}\cdot T_{\mathrm{s}}+\left(1-P_{\mathrm{s},1}-P_{\mathrm{idle},1}\right)T_{\mathrm{c}}}\\ S_2=\dfrac{\tau_2\cdot(1-\tau_1)\cdot E[P]}{T_{\mathrm{slot}}\cdot P_{\mathrm{idle},2}+P_{\mathrm{s},2}\cdot T_{\mathrm{s}}+\left(1-P_{\mathrm{s},2}-P_{\mathrm{idle},2}\right)T_{\mathrm{c}}}\end{cases}\tag{5.2.3}$$

注意到$P_{\mathrm{idle}}=P_{\mathrm{idle},1}=P_{\mathrm{idle},2}$、$P_{\mathrm{s}}=P_{\mathrm{s},1}=P_{\mathrm{s},2}$、$\tau=\tau_1=\tau_2$，显然$S_1=S_2$。

由$S=\sum_{n\in\mathcal{N}}S_n$，有

$$S=\frac{2\tau\cdot(1-\tau)\cdot E[P]}{T_{\mathrm{slot}}\cdot P_{\mathrm{idle}}+P_{\mathrm{s}}\cdot T_{\mathrm{s}}+(1-P_{\mathrm{s}}-P_{\mathrm{idle}})\cdot T_{\mathrm{c}}}\tag{5.2.4}$$

注意到$P_{\mathrm{s}}=2\tau\cdot(1-\tau)$，显然式(5.2.4)得到的系统吞吐量与简单模型中式(4.3.5)一致，

这也验证了这两个模型的一致性。

5.3. 模型理论求解

首先明确问题一给出的参数如表 5-1 所示：

参数名称	值
PHY 头时长	13.6μs
ACK 时长T_{ACK}	32μs
SIFS 时长T_{SIFS}	16μs
DIFS 时长T_{DIFS}	43μs
SLOT 时长T_{slot}	9μs
ACK 超时时间T_{TO}	63μs
最小窗口W_0	16
最大窗口W_m	1024
最大重传次数r	32
MAC 头长度L_{MAC}	30 Bytes
有效载荷长度L_{PLD}	1500 Bytes
RTS 帧长度L_{RTS}	20 Bytes
CTS 帧长度L_{CTS}	14 Bytes
物理层速率B	455.8 Mbps

表 5-1 问题一涉及到的参数列表

对于问题一的数值求解，首先由最大窗口和最小窗口，可以求得最大退避阶数$m = 6$。之后联立式(4.2.9)、式(4.2.10)和式(4.2.11)，将$W_0 = 16, m = 6, r = 32, N = 2$代入公式计算，联立方程后得到仅含有 p 的多项式方程，使用牛顿法即可快速求得该多项式方程在范围$p \in (0,1)$上的实根，由式(4.2.10)可求出 τ 的值，将 τ 代入式(4.3.1)与式(4.3.2)中分别求得 P_{idle} 和 P_{s}，最后由式(4.3.4)，可以得到系统吞吐量。

通过 MATLAB 进行数值计算，可以得到节点在某个时隙发送包发生冲突概率的可行解 $p = 10.46\%$，在某时隙信道空闲概率$P_{\mathrm{idle}} = 80.17\%$。最终得到系统的理论总吞吐量为 67.1744 Mbps，理论结果如下表所示：

理论建模各项参数	理论建模各项值
p	10.46%
P_{idle}	80.17%
S	67.1744 Mbps

表 5-2 问题一理论建模各项参数结果

在此基础上，考虑 RTS/CTS 机制对系统吞吐量的影响。由式(4.3.8)可以计算启用 RTS/CTS 机制式的系统总吞吐量S_{RTS}。

经过求解可得在启用 RTS/CTS 机制时，系统的理论吞吐为 50.6569Mbps，显著低于不启用 RTS/CTS 时的吞吐量。究其原因，是因为此模式下冲突的概率比较低，启用 RTS/CTS 时虽然可以一定程度上减少碰撞，但是减小碰撞的收益低于发送 RTS/CTS 帧所增加的时间开销。

事实上，不妨求解方程$S_{\mathrm{RTS}} > S$，并将长度参数代入式(4.3.8)，可得

$$P_{\mathrm{s}} < \frac{1510 \times 8}{59.2 \cdot B + 1544 \times 8} \tag{5.3.1}$$

其中B的单位为 Mbps，代入物理层速率$B = 455.8$ Mbps时，可以解得$P_{\mathrm{s}} < 30.71\%$时启用 RTS/CTS 机制吞吐量更优，反之不启用时吞吐量更优。

5.4. 模型仿真验证

本文基于 MATLAB 自行编写了一个仿真器，该仿真器采用事件驱动模型，将每个节点抽象为了类。每个类中包含的退避、等待、传输、冲突和超时重传等状态，把各个状态间的切换当成事件，并采用事件驱动的方式对整个系统进行仿真模拟，实现了对 WLAN 中各节点采用二进制退避算法的分布式信道接入场景下的通信性能进行仿真模拟，并且实现了对各种事件的可视化展示，详细代码见附录 1。该仿真器有着良好的封装、灵活的参数配置选项的功能，只需对主脚本进行几行配置代码的修改，就能够实现对各种场景的仿真模拟。下面将基于我们编写的仿真器对问题一的 2 BSS 场景进行仿真验证，实现对理论结果的验证与支撑。

问题一对建立在第四章基本模型上的 2 BSS 系统进行仿真验证，本文通过 MATLAB 仿真器对 2 BSS 系统中两个 AP 吞吐量以及系统总吞吐量进行了统计。如图 5-1 所示，其中红线代表 AP1 吞吐量，绿线代表 AP2 吞吐量，黄线代表系统总吞吐量。本题的吞吐量统计方式是：统计节点在一段间隔内收到 ACK 的数量，乘以有效载荷，再除以事件间隔，即得到每个 AP 的吞吐量，求和即为系统总吞吐量。

经过仿真可得在 0 至2×10^6μs内，系统总平均吞吐量为 65.4638 Mbps，与模型的数值解相差 2.5%，可以证明本题理论模型的正确性，以及本文仿真器的准确性和稳定性。

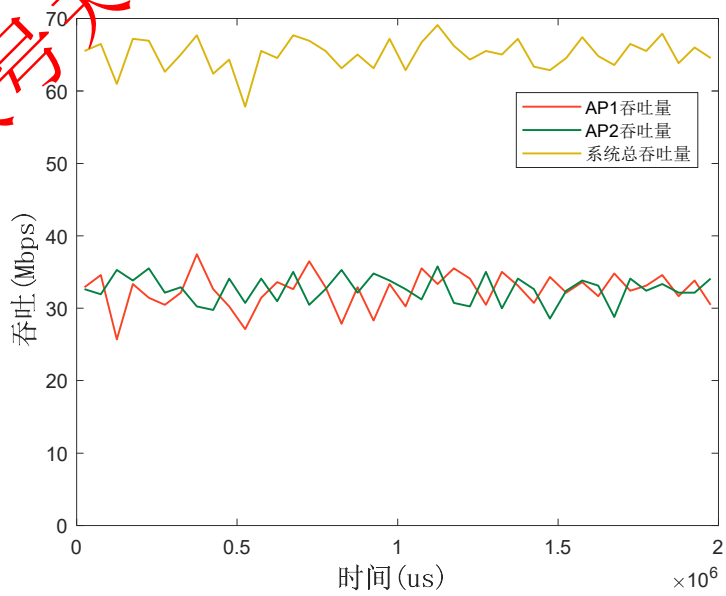

图 5-1 问题一 2 BSS 系统的吞吐量

为了进一步理解通信的细节，利用本文仿真器对事件的可视化展示功能，本文绘制了仿真事件的结果如图 5-2 所示。为了让结果更清晰，图中只截出$0-10000\mu s$发生的事件，其中黄线代表节点发送数据，绿线代表节点收到 ACK，红线代表节点超时，分别用 1，2，3 表示。由仿真事件图可知，在绝大部分时间，节点发送数据后能够收到 ACK 消息，本题计算仿真结果中收到 ACK 的次数占总发送次数的比值，得到节点在某个时隙发送包发生冲突概率为 11.03%，与理论值相差 0.57%，符合仿真实验预期，证明问题一理论模型的正确性以及仿真的完整性和准确性。

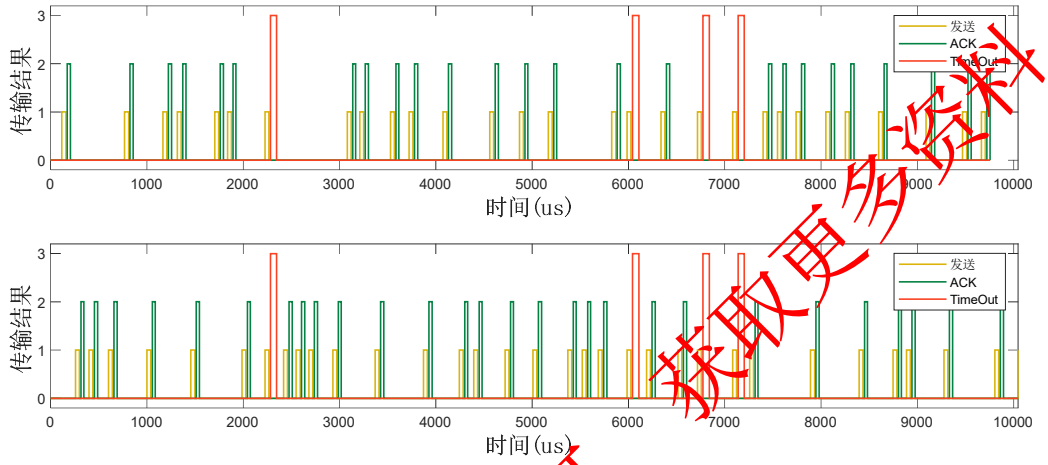

图 5-2 问题一仿真事件图

6. 问题二的分析与求解

6.1. 问题分析

问题二考虑 BSS 间可以互听，且在 AP 并发时数据传输成功的场景。对该场景下的 2 BSS 系统进行建模。

首先明确此问题中的 BSS 间可以互听，因此不存在隐蔽站问题，且在 AP 并发时数据传输将失败，因此存在暴露站问题。其对应于 4.2 和 4.3 节中涉及到的简单场景下的冲突概率和吞吐量建模模型。

问题二与问题一的区别在于，并发传输数据时两个终端接收到数据的 SIR 较高，传输都能够成功。由于不会出现传输失败的情况，当信道非空闲时，数据一定传输成功，对应信道传输失败概率$P_c=0$。

问题二的情景对应于网络中暴露终端的概念，在本节中同样引入 RTS/CTS 机制，并讨论其对暴露终端问题的影响。同样的，在本题中假设传播时延足够小，可以忽略。

根据以上分析和 4.2 和 4.3 节中提到的理论模型，由式(4.2.9)、式(4.2.10)和式(4.2.11)可以得到某时刻节点发包的冲突概率。根据式(4.3.5)，可以得到系统吞吐量，根据式(4.3.8)，可以得到开启 RTS/CTS 模式下的系统吞吐量。

6.2. 模型建立

直观地看，问题二的模型相当于在问题一的模型中令 $p=0$ 以及 $P_c=0$ 得到的结果。本文在 4.5 节给出了一个一般化的模型，可以彻底地解决问题二。

我们将 4.5 节中一般化的模型代入本问题的条件进行推导，最终将推导出除了$p=0$和

$P_c = 0$以外均与 4.2、4.3 节一致的结论，从而展示这个一般化模型的用途与正确性。

对于问题二而言，全网只有两个节点，分别记为 1、2。它们两者互听但不互干扰，因此有$\mathbb{D}_1 = \emptyset, \mathbb{D}_2 = \emptyset, \mathbb{E}_1 = \{2\}, \mathbb{E}_2 = \{1\}, \mathbb{F}_1 = \emptyset, \mathbb{F}_2 = \emptyset$，本问题中的丢包率$P_e = 0$。

对于每一个节点，由式(4.5.3)，代入$P_e = 0, \mathbb{D}_1 = \emptyset, \mathbb{D}_2 = \emptyset, \mathbb{F}_1 = \emptyset, \mathbb{F}_2 = \emptyset$可以得到：

$$\begin{cases} p_1 = 1 - 1 = 0 \\ p_2 = 1 - 1 = 0 \end{cases} \tag{6.2.1}$$

对于$P_{\text{idle},n}$，由式(4.5.5)，代入$\mathbb{D}_1 = \emptyset, \mathbb{D}_2 = \emptyset, \mathbb{E}_1 = \{2\}, \mathbb{E}_2 = \{1\}$，可见两者的$P_{\text{idle},n}$均为$(1-\tau_1)(1-\tau_2)$，代入$\tau = \tau_1 = \tau_2$可知与式(4.3.1)一致，即$P_{\text{idle}} = P_{\text{idle},1} = P_{\text{idle},2}$。

对于$P_{s,n}$，由式(4.5.7)，注意到该系统中两节点均互不干扰，故$\lambda_{1,2} = 1$，此时$P_{s,n}$退化为$P_{s,n} = \sum_{i\in\{n\}\cup\mathbb{D}_n\cup\mathbb{E}_n}\left(\tau_j \cdot (1-p_j)\right) - \tau_1\tau_2(1-p_1)(1-p_2)$，代入$P_e = 0, \mathbb{D}_1, \mathbb{D}_2, \mathbb{E}_1, \mathbb{E}_2$，有：

$$\begin{cases} P_{s,1} = \tau_1 + \tau_2 - \tau_1\tau_2 \\ P_{s,2} = \tau_1 + \tau_2 - \tau_1\tau_2 \end{cases} \tag{6.2.2}$$

由$P_{c,n} = 1 - P_{\text{idle},n} - P_{s,n}$，显然$P_{c,1} = P_{c,2} = 0$，这与预期一致。

此时对于每个节点而言，由式(4.5.8)，可得单个节点的吞吐量为：

$$\begin{cases} S_1 = \dfrac{\tau_1 \cdot (1-\tau_2) \cdot E[P]}{T_{\text{slot}} \cdot P_{\text{idle},1} + P_{s,1} \cdot T_s + (1 - P_{s,1} - P_{\text{idle},1})T_c} \\ S_2 = \dfrac{\tau_2 \cdot (1-\tau_1) \cdot E[P]}{T_{\text{slot}} \cdot P_{\text{idle},2} + P_{s,2} \cdot T_s + (1 - P_{s,2} - P_{\text{idle},2})T_c} \end{cases} \tag{6.2.3}$$

注意到$P_{\text{idle}} = P_{\text{idle},1} = P_{\text{idle},2}$、$P_s = P_{s,1} = P_{s,2}$、$\tau = \tau_1 = \tau_2$，显然$S_1 = S_2$。

由$S = \sum_{n\in\mathcal{N}} S_n$，有

$$S = \frac{2\tau \cdot (1-\tau) \cdot E[P]}{T_{\text{slot}} \cdot P_{\text{idle}} + P_s \cdot T_s + (1 - P_s - P_{\text{idle}}) \cdot T_c} \tag{6.2.4}$$

注意到$P_s = 2\tau \cdot (1-\tau)$，显然式(6.2.4)得到的系统吞吐量与简单模型中式(4.3.5)一致。

6.3. 模型理论求解

问题二中的物理层速率为 275.3Mbps，其余参数与问题一相同，参见表 6-3。

对于问题二的数值求解，首先由最大窗口和最小窗口，可以求得最大退避阶数$m = 6$。之后联立式(4.2.9)、式(4.2.10)和式(4.2.11)，将$W_0 = 16, m = 6, r = 32, N = 2$代入公式计算，联立方程后得到仅含有 p 的多项式方程，由式(4.2.10)可求出 τ 的值，将 τ 代入式(4.3.1)与式(4.3.2)中分别求得 P_{idle} 和 P_s，最后由式(4.3.5)，可以得到系统吞吐量。

通过 MATLAB 进行数值计算，可以得到节点在某个时隙发送包发生冲突概率的可行解 $p = 0$，在某时隙信道有人占用的概率$P_{\text{idle}} = 77.85\%$。最终得到系统的理论吞吐为 62.26Mbps，理论结果如下表所示：

理论建模各项参数	理论建模各项值
p	0
P_{idle}	77.85%
S	62.26Mbps

表 6-3 问题二理论参数值

考虑 RTS/CTS 机制对系统吞吐量的影响。经过求解可得在启用 RTS/CTS 机制时，系统的理论吞吐为 50.3148Mbps，显著低于不启用 RTS/CTS 时的吞吐量。究其原因，是因为此场景下数据包传输一定成功，启用 RTS/CTS 不能减少碰撞，却带来了发送 RTS/CTS 帧所增加的时间开销。开启 RTS/CTS 能否提升系统吞吐量主要取决于信道繁忙时数据发送的成功概率P_s。在此物理层速率和网络拓扑场景下，当 P_s 高于 42.16%时，不启用 RTS/CTS 系统吞吐量更高。当 P_s 低于 42.16%时，启用 RTS/CTS 系统吞吐量更高。

6.4. 模型仿真验证

问题二对建立在第四章基本模型上的 2 BSS 系统进行仿真验证，本文通过 MATLAB 仿真器对 2 BSS 系统中两个 AP 吞吐量以及系统总吞吐量进行了统计。如图 6-1 所示，其中红线代表 AP1 吞吐量，绿线代表 AP2 吞吐量，黄线代表系统总吞吐量。本题的吞吐量统计方式是：统计节点在一段间隔内收到 ACK 的数量，乘以有效载荷，再除以事件间隔，即得到每个 AP 的吞吐量，求和即为系统总吞吐量。

在问题二模型的基础上，增加无冲突这一条件后，系统总吞吐量并不会陡然增加，这是由于节点发送的数据包中除了有效载荷外，还有物理层头和 MAC 层头，同时 ACK、DIFS、SIFS 和退避时间占用较多的时间，使得吞吐量的最大值受到限制。此外，与第一问相比，物理层速率也下降为了 275.3Mbps。通过模型仿真验证，得到在问题二的模型下，在 0 至$2 \times 10^6 \mu s$内，系统总平均吞吐量为 69Mbps。证明模型的正确性以及本文仿真器的准确性和稳定性。

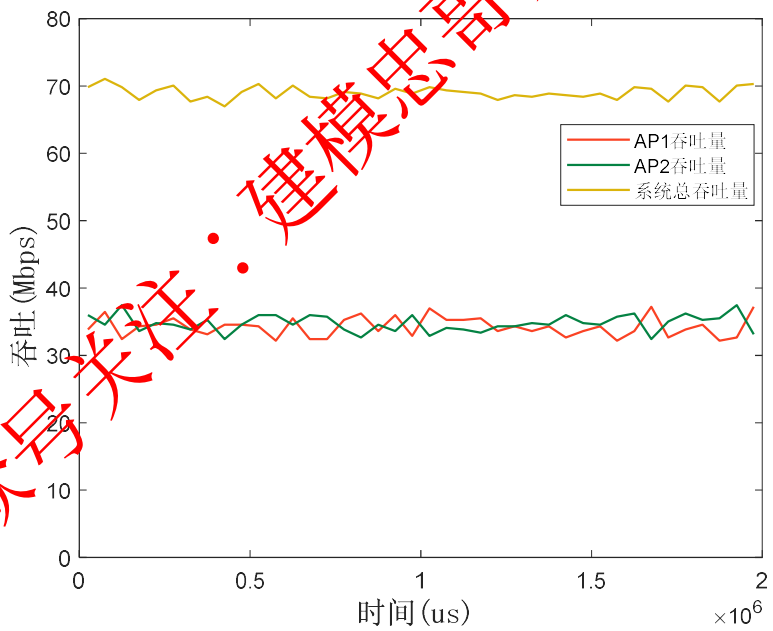

图 6-1 问题二 2BSS 系统的吞吐量

同时本文绘制了仿真事件的结果如图 6-2 所示。由于问题二描述中两节点并发传输不会失败，仿真结果与其一致，节点发送数据后都会收到 ACK 消息，没有超时现象，并且可以观察到图 6-2 中存在两个节点同时发送数据的事件，并且同时发送数据后仍然收到 ACK 消息，即系统碰撞概率$p = 0$，与理论值相同，符合仿真实验预期，证明问题二理论模型的正确性以及仿真的完整性和准确性。

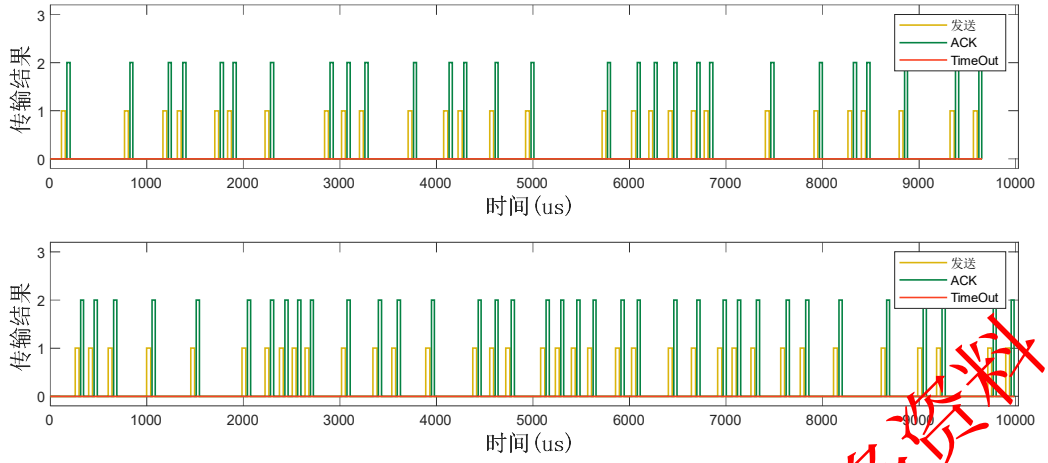

图 6-2 问题二仿真事件图

7. 问题三的分析与求解

7.1. 问题分析

问题三考虑 BSS 间不可以互听，且在 AP 并发时数据传输失败的场景，并认为在单个 AP 发送时也存在 $P_e = 10\%$ 的丢包。对该场景下的 2 BSS 系统进行建模。

首先明确此问题中的 BSS 间不可以互听，因此存在隐蔽站问题，且在 AP 并发时数据传输将失败，因此不存在暴露站问题。此场景实际为隐藏终端问题。这对应了 4.4 节提到的隐藏终端场景下的分析模型，并且单个 AP 发送时也存在 $P_e = 10\%$ 的丢包，这对应了 4.4 节中的冲突概率和吞吐量分析模型。

此外，考虑到此问题属于隐藏终端问题。对于隐藏终端问题[3]，首先需要考虑的是引入 RTS/CTS 机制，接收方发送 CTS 包要求隐蔽站等待来避免冲突。其次，在 4.4 节对隐蔽站的分析，本文将覆盖站和隐蔽站分开求解各自的冲突概率 τ 与 κ，根据覆盖站和隐蔽站的数量 N_{cov} 与 N_{hid} 求解系统的冲突概率 p，依据 4.4 节对隐藏终端的分析求出系统的总吞吐量 S。

最后问题三需要考虑到 $P_e = 10\%$ 的丢包率，这主要会影响到冲突概率 p_n 的计算，具体计算方法如式(4.5.3)所示。

7.2. 模型建立

问题三的模型建立在 4.4 节中完成，得到了最大重传次数约束条件下的系统归一化的吞吐量 S 及开启 RTS/CTS 模式下的系统吞吐量 S_{RTS}。

由于在 4.5 节处我们推导出了一个更一般的模型，该模型可以用于解决所有场景下的问题。此处我们将基于这个一般化的模型代入本问题的条件进行推导，最终将推导出与 4.5 节一致的结论，从而展示这个一般化模型的用途与正确性。

对于问题三而言，全网只有两个节点，分别记为 1、2。它们两者不互听但互干扰，因此有 $\mathbb{D}_1 = \emptyset, \mathbb{D}_2 = \emptyset, \mathbb{E}_1 = \emptyset, \mathbb{E}_2 = \emptyset, \mathbb{F}_1 = \{2\}, \mathbb{F}_2 = \{1\}$，本问题中的丢包率 $P_e = 10\%$。

对于每一个节点，由式(4.5.3)，代入 $P_e = 0, \mathbb{D}_1 = \emptyset, \mathbb{D}_2 = \emptyset, \mathbb{F}_1 = \{2\}, \mathbb{F}_2 = \{1\}$ 可以得到：

$$\begin{cases} p_1 = 1 - (1 - P_e)(1 - \kappa_2) \\ p_2 = 1 - (1 - P_e)(1 - \kappa_1) \end{cases} \tag{7.2.1}$$

其中κ_k的表达式见式(4.5.4)，κ_k是关于p_k的有理多项式。

显然p_1和p_2的表达式具有交换对称性，并且节点 1、2 的拓扑也具有交换对称性，可以得到$p_1 = p_2$。因此我们不必直接求解这个二元多项式方程组，直接代入$p_1 = p_2$就退化为了与式(4.4.3)一致的一元多项式方程组了，这也验证了我们的推广模型与简单模型的一致性。

对于$P_{\text{idle},n}$，由式(4.5.5)，代入$\mathbb{D}_1 = \emptyset, \mathbb{D}_2 = \emptyset, \mathbb{E}_1 = \emptyset, \mathbb{E}_2 = \emptyset$可得：

$$\begin{cases} P_{\text{idle},1} = (1-\tau_1) \\ P_{\text{idle},2} = (1-\tau_2) \end{cases} \tag{7.2.2}$$

由于本问题中有$p_1 = p_2$，自然有$\tau = \tau_1 = \tau_2$，可知与 4.4 节的$P_{\text{idle}} = (1-\tau)^{N_{\text{cov}}}$一致，即$P_{\text{idle}} = P_{\text{idle},1} = P_{\text{idle},2}$。

对于$P_{\text{s},n}$，由式(4.5.7)，注意到该系统中两节点均互相干扰，故$\lambda_{1,2} = 0$，此时$P_{\text{s},n}$退化为$P_{\text{s},n} = \sum_{i\in\{n\}\cup\mathbb{D}_n\cup\mathbb{E}_n}\left(\tau_j\cdot(1-p_j)\right)$，代入$\mathbb{D}_1, \mathbb{D}_2, \mathbb{E}_1, \mathbb{E}_2, \mathbb{F}_1, \mathbb{F}_2$，有：

$$\begin{cases} P_{\text{s},1} = \tau_1\cdot(1-p_1) = \tau_1(1-P_{\text{e}})(1-\kappa_2) \\ P_{\text{s},2} = \tau_2\cdot(1-p_2) = \tau_2(1-P_{\text{e}})(1-\kappa_1) \end{cases} \tag{7.2.3}$$

代入$\kappa = \kappa_1 = \kappa_2$可知与式(4.4.)一致，即$P_{\text{s}} = P_{\text{s},1} = P_{\text{s},2}$。

此时对于每个节点而言，由式(4.5.8)，可得单个节点的吞吐量为：

$$\begin{cases} S_1 = \dfrac{\tau_1\cdot(1-P_{\text{e}})(1-\kappa_2)\cdot E[P]}{T_{\text{slot}}\cdot P_{\text{idle},1} + P_{\text{s},1}\cdot T_{\text{s}} + (1-P_{\text{s},1}-P_{\text{idle},1})T_{\text{c}}} \\ S_2 = \dfrac{\tau_2\cdot(1-P_{\text{e}})(1-\kappa_1)\cdot E[P]}{T_{\text{slot}}\cdot P_{\text{idle},2} + P_{\text{s},2}\cdot T_{\text{s}} + (1-P_{\text{s},2}-P_{\text{idle},2})T_{\text{c}}} \end{cases} \tag{7.2.4}$$

注意到$P_{\text{idle}} = P_{\text{idle},1} = P_{\text{idle},2}$、$P_{\text{s}} = P_{\text{s},1} = P_{\text{s},2}$、$\tau = \tau_1 = \tau_2, \kappa = \kappa_1 = \kappa_2$，显然$S_1 = S_2$。

由$S = \sum_{n\in\mathcal{N}} S_n$，有

$$S = \frac{2\tau\cdot(1-\kappa)(1-P_{\text{e}})\cdot E[P]}{T_{\text{slot}}\cdot P_{\text{idle}} + P_{\text{s}}\cdot T_{\text{s}} + (1-P_{\text{s}}-P_{\text{idle}})\cdot T_{\text{c}}} \tag{7.2.5}$$

显然式(7.2.5)得到的系统吞吐量与只考虑隐蔽站模型中式(4.4.5)一致，这也验证了这两个模型的一致性。

7.3. 模型理论求解

首先将式(4.2.10)、式(4.4.2)和式(4.4.3)联立，得到只含有冲突概率 p 的多项式方程，求解出 p 的数值解，同时得到因隐蔽站的脆弱时期导致的 κ，以及马尔可夫链中$b_{0,0}$的值。进而可以根据题目中的参数要求，由式(4.5.8)仿真不同吞吐量，不同最小窗口和最大窗口，最大重传次数场景下的理论吞吐量。其中，物理层速率，最小窗口和最大窗口及最大重传次数的可选范围如表 7-1 所示。

$W_{\min}$	$W_{\max}$	最大重传次数	物理层速率
16	1024	6	455.8Mbps
32	1024	5	455.8Mbps
16	1024	32	455.8Mbps
16	1024	6	286.8Mbps
32	1024	5	286.8Mbps
16	1024	32	286.8Mbps

16	1024	6	158.4Mbps
32	1024	5	158.4Mbps
16	1024	32	158.4Mbps

表 7-1 最小窗口和最大窗口，最大重传次数的取值范围

通过 MATLAB 数值计算，最终得到在不同物理层速率，最小窗口和最大窗口，最大重传次数的条件下，系统的吞吐量性能和冲突概率如图 7-1 和图 7-2 所示。

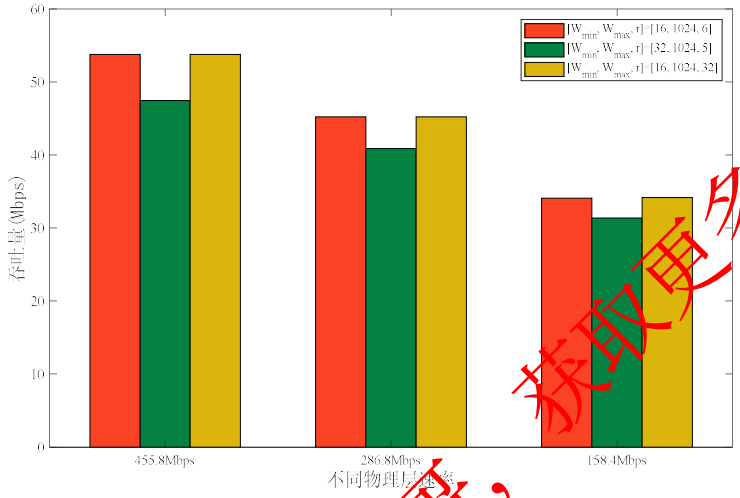

图 7-1 不同物理层速率下理论系统吞吐量

从图 7-1 中可以得知，在此场景下，物理层速率是影响系统吞吐量的决定性因素。随着物理层速率下降，系统成功发送单个数据包的期望时间增加，系统吞吐量下降。最小窗口的大小也对系统的吞吐量也有一定的影响，红色曲线和绿色曲线相比，最小窗口增大，吞吐量减小，这是因为退避窗口增加会提高的节点处于退避状态时间的期望，这会一定程度上降低系统吞吐量。此外，比较红色曲线和黄色曲线，最大重传次数对系统吞吐量的影响不明显，这可能是因为网络中节点较少，发生多次碰撞的可能性较低，很难达到最大重传次数，因此其对系统吞吐量的影响不明显。

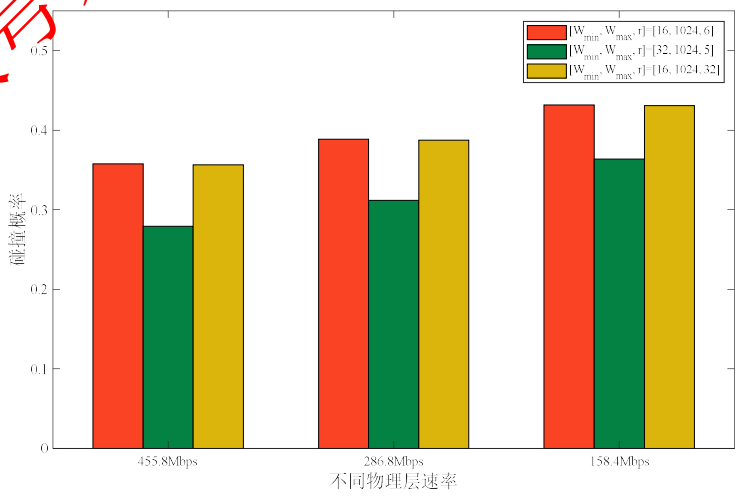

图 7-2 不同物理层速率下的理论系统碰撞概率

从图 7-2 中可以得知，在此场景下，物理层速率也是影响系统碰撞概率的决定性因素。随着物理层速率下降，系统碰撞概率增加，系统吞吐量下降。最小窗口的大小也对系统的吞吐量也有一定的影响，红色曲线和绿色曲线相比，最小窗口较小，碰撞概率较大。这是因为退避窗口增加会提高的节点处于退避状态时间的期望，这会一定程度上减小碰撞概率。此外，比较红色曲线和黄色曲线，最大重传次数对碰撞概率的影响不明显，这可能是因为网络中节点较少，发生多次碰撞的可能性较低，很难达到最大重传次数，因此最大重传次数对系统碰撞概率的影响不明显。

由于此场景涉及到了隐藏终端问题，由于启用 RTS/CTS 机制在系统碰撞概率较大时可以提升系统的吞吐量。因此考虑对启用 RTS/CTS 机制的场景进行仿真验证，此时系统吞吐量如图 7-3 所示。

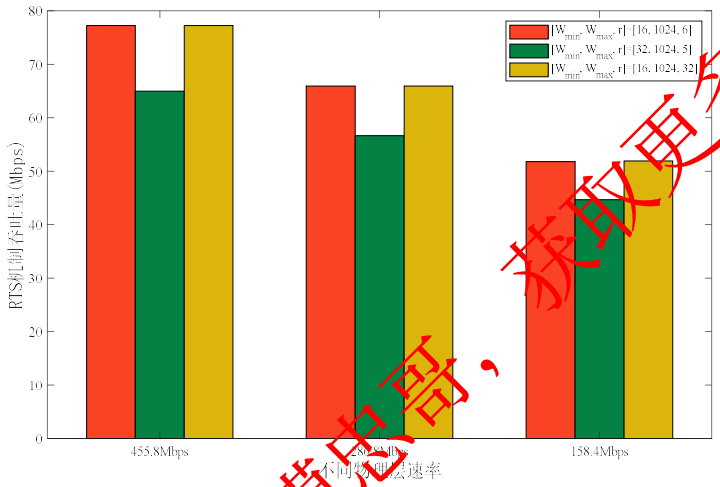

图 7-3 不同物理层速率下启用 RTS/CTS 时的系统吞吐量

从图 7-3 中可以得知，在此场景下，启用 RTS/CTS 可以显著提升系统的吞吐量。这是因为在隐藏终端场景下，节点发生碰撞的概率较高，启用 RTS/CTS 时减小碰撞的收益高于发送 RTS/CTS 帧所增加的时间开销。

7.4. 模型仿真验证

通过基于 MATLAB 的仿真器对上述结果进行仿真验证。仿真时间为$2 \times 10^6 \mu s$，本题通过改变竞争窗口和最大重传次数对不同物理层速率，最小窗口和最大窗口及最大重传次数场景分别进行仿真，仿真结果如下。

7.4.1. 物理层速率为 455.8Mbps 下场景

首先通过本文的仿真器，对$B = 455.8\text{Mbps}, W_{\min} = 16, W_{\max} = 1024, r = 32$的场景进行了仿真验证。其仿真事件图和系统吞吐量如图 7-4 和 7-5 所示。

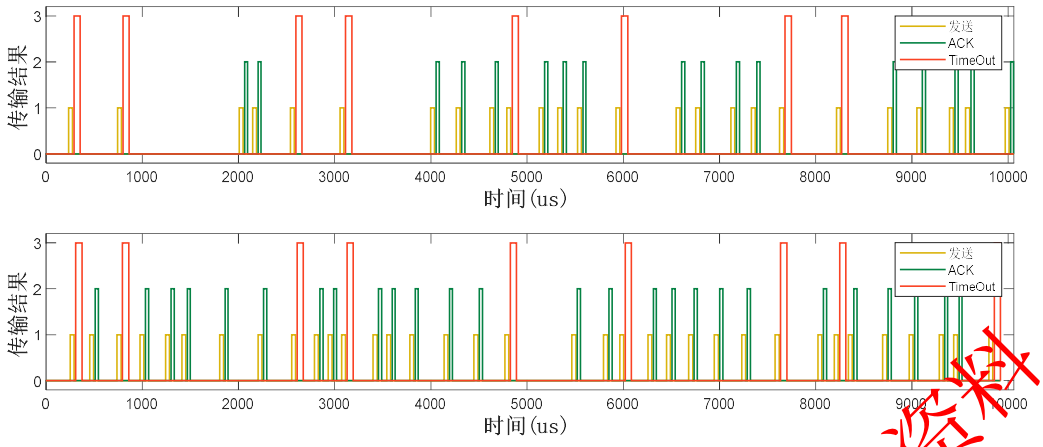

图 7-4 问题三B = 455.8Mbps时仿真事件图

由仿真事件图 7-4 可知，在隐蔽站存在的情况下 AP1 与 AP2 不互听，使得在 AP1 发送数据时 AP2 也能够发送数据，产生冲突导致双方多次超时，大大降低了系统的吞吐量，增加了系统的碰撞概率，本题通过仿真器得出当前情况下的系统碰撞概率为 34.92%，理论值为 35.46%，吞吐碰撞概率与理论值相差 0.5%左右，证明了模型的正确性和仿真器的有效性。

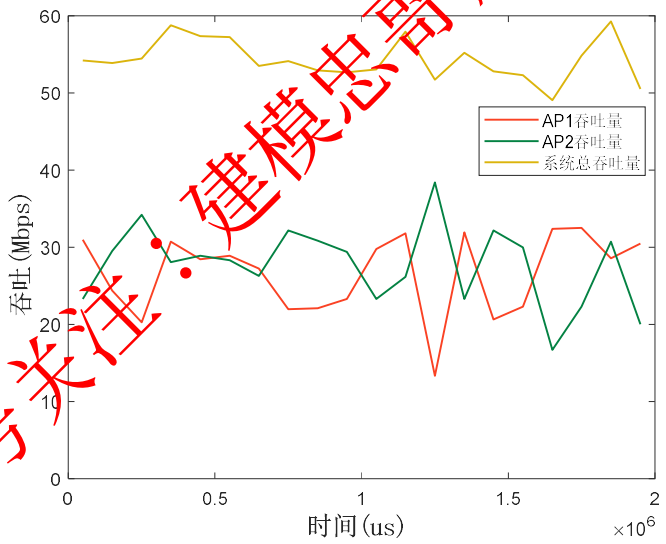

图 7-5 问题三B = 455.8Mbps时系统吞吐量图

由于碰撞概率达到 34.92%，由图 7-5 可以观察到由于隐藏终端问题的存在，AP1 与 AP2 的吞吐量随时间变化曲线非常剧烈。当 AP1 在发送数据时，AP2 不能感知到 AP1 的存在，因此发送数据时出现冲突。但是系统的总吞吐量基本保持稳定，在仿真时间段内，系统的平均吞吐量为 54.3Mbps，理论值为 53.7708Mbps，仿真的平均吞吐量与理论值相比相差 1%，在合理的误差范围内。

7.4.2. 物理层速率为 286.8Mbps 和 158.4Mbps 场景

根据表 7-1 中的参数，通过本文基于 MATLAB 实现的仿真器得到系统的吞吐量和碰撞概率如图 7-6 和图 7-7 所示。

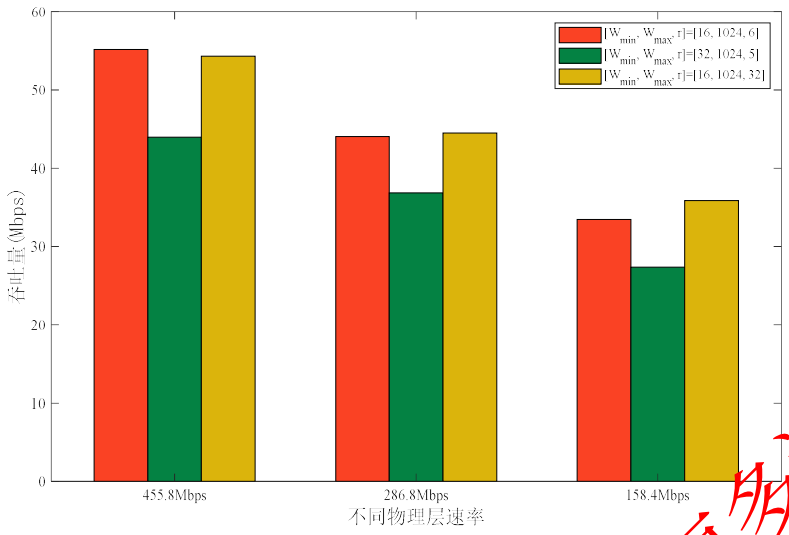

图 7-6 不同条件下场景三的系统吞吐量对比

从图 7-6 中可以得知，在此场景下，系统吞吐量主要受到物理层速率和最小窗口的影响。最小窗口的大小也对系统的吞吐量也有一定的影响，红色曲线和绿色曲线相比，最小窗口增大，吞吐量减小，这是因为退避窗口增加会提高的节点处于退避状态时间的期望，这会一定程度上降低系统吞吐量。此外，比较红色曲线和黄色曲线，最大重传次数对系统吞吐量的影响不明显，这可能是因为网络中节点较少，发生多次碰撞的可能性较低，很难达到最大重传次数，因此其对系统吞吐量的影响不明显。

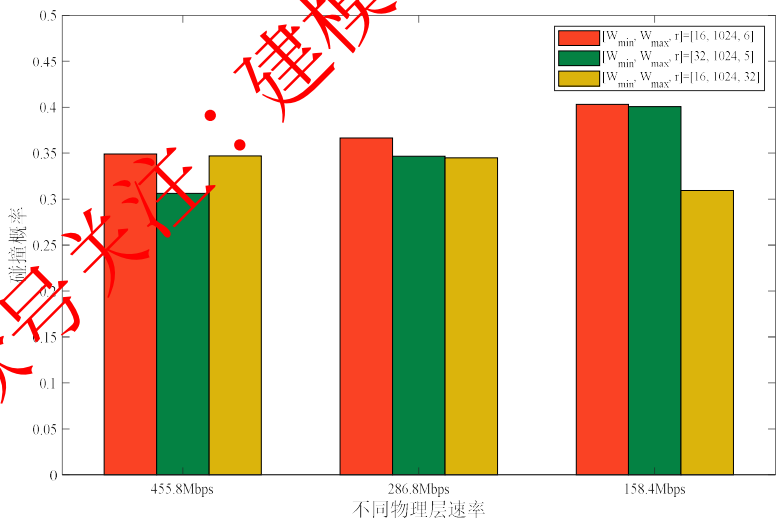

图 7-7 不同条件下场景三的碰撞概率对比

从图 7-7 中可以得知，在此场景下，随着物理层速率下降，系统碰撞概率增加。最小窗口的大小也对系统的吞吐量也有一定的影响，红色曲线和绿色曲线相比，最小窗口较小，碰撞概率较大。这是因为退避窗口增加会提高的节点处于退避状态时间的期望，这会一定程度上减小碰撞概率。此外，比较红色曲线和黄色曲线，物理层速率较高时，重传次数对碰撞概率的影响不明显，这可能是此时碰撞的可能性较低，很难达到最大重传次数。而在物理层速率较低时，重传次数对碰撞概率有一定影响，这是因为随着物理层速率降低，碰

撞重传的可能性较大，达到最大窗口后，重传次数越多，退避等待的时隙越长，系统碰撞概率越低。

将图 7-3 和图 7-6 中的不同场景下的系统吞吐量整理为表 7-2，可以得到不同场景下的系统吞吐量的理论值和仿真值的对比。

物理层速率	W_{min}	最大重传次数	理论吞吐量 Mbps	仿真吞吐量 Mbps	误差
455.8Mbps	16	6	53.77	55.17	2.5%
	32	5	47.47	43.99	7%
	16	32	53.77	54.3	1%
286.8Mbps	16	6	45.21	44.05	2.5%
	32	5	40.89	36.86	9.8%
	16	32	45.24	44.49	1.7%
158.4Mbps	16	6	34.11	33.46	1.9%
	32	5	31.35	27.35	12.7%
	16	32	34.18	35.88	4.7%

表 7-2 不同场景下的理论吞吐量和仿真吞吐量对比

根据问题三的要求，本文在仿真器中仿真了 2 BSS 系统在 3 种不同物理层速率下，改变最小竞争窗口和最大重传次数得到不同状态下系统的吞吐量和碰撞概率，共 9 组数据如上表所示，根据表格可知，理论吞吐量和仿真吞吐量的平均误差在 5%以内，证实了理论模型的正确性和仿真系统的有效性。

8. 问题四的分析与求解

8.1. 问题分析

问题要求考虑 3 BSS 的场景，假设 BSS1 与 BSS3 不互听，BSS2 与两者都互听，且在 AP1 和 AP3 并发时数据可以成功传输，对 3 BSS 系统进行建模。

首先明确此问题中的 BSS1 与 BSS3 不互听，BSS2 与两者都互听，但是 AP1 和 AP3 并发时数据可以成功传输，因此不存在隐藏终端问题。由于此问题涉及到的场景包含了节点间能否互听和节点间是否干扰两种场景，因此我们采用我们第四章提出的统一的复杂场景下的单节点冲突概率和吞吐量模型。将单节点吞吐量求和，即可得到系统总吞吐量模型。

由问题四的题目描述，AP1 与 AP2 之间互听，AP2 与 AP3 之间互听，AP1 与 AP3 之间不互听。题目只说了 AP1 与 AP3 之间不会互相干扰，但是没有提到 AP1 与 AP2 之间、AP2 与 AP3 之间会不会互相干扰，因此此处分为四种情况进行讨论。

1. AP1 与 AP2 之间会互相干扰，AP2 与 AP3 之间会互相干扰。
2. AP1 与 AP2 之间会互相干扰，AP2 与 AP3 之间不会互相干扰。
3. AP1 与 AP2 之间不会互相干扰，AP2 与 AP3 之间会互相干扰。
4. AP1 与 AP2 之间不会互相干扰，AP2 与 AP3 之间不会互相干扰。

8.2. 模型建立

最普适的模型在 4.5 节中完成，它适用于本题目中的所有情形，自然也包含了问题四。让我们将问题四的条件代入 4.5 节的普适模型，得到关于问题四的特解。

下面对 8.1 节提到的四种可能情况分别分析。

8.2.1. 情况一：AP1 与 AP2 之间会互相干扰，AP2 与 AP3 之间会互相干扰

对于问题三而言，全网只有三个节点，分别记为 1、2、3。相邻节点互听且互干扰，因此有$\mathbb{D}_1=\{2\},\mathbb{D}_2=\{1,3\},\mathbb{D}_3=\{2\},\mathbb{E}_1=\emptyset,\mathbb{E}_2=\emptyset,\mathbb{E}_3=\emptyset,\mathbb{F}_1=\emptyset,\mathbb{F}_2=\emptyset,\mathbb{F}_3=\emptyset$，本问题中的丢包率$P_{\mathrm{e}}=0\%$，故不考虑丢包问题。

对于每一个节点，由式(4.5.3)，代入$P_{\mathrm{e}}=0,\mathbb{D}_k,\mathbb{F}_k,k\in[1,3]$可以得到：

$$\begin{cases}p_1=1-(1-\tau_2)=\tau_2\\p_2=1-(1-\tau_1)(1-\tau_3)\\p_3=1-(1-\tau_2)=\tau_2\end{cases}\tag{8.2.1}$$

其中$\tau_n=b_{0,0,n}[1-(p_n)^{r+1}]/(1-p_n)$，$b_{0,0,n}$见式(4.5.2)，$\tau_n$是关于$p_n$的有理多项式。

显然p_1和p_3的表达式具有交换对称性，并且节点 1、3 的拓扑也具有交换对称性，可以得到$p_1=p_3$，此时该 3 元多项式方程组被约化为了 2 元多项式方程组。

对于$P_{\mathrm{idle},n}$，由式(4.5.5)，代入$P_{\mathrm{e}}=0,\mathbb{D}_k,\mathbb{E}_k,k\in[1,3]$可得：

$$\begin{cases}P_{\mathrm{idle},1}=(1-\tau_1)(1-\tau_2)\\P_{\mathrm{idle},2}=(1-\tau_2)(1-\tau_1)(1-\tau_3)\\P_{\mathrm{idle},3}=(1-\tau_3)(1-\tau_2)\end{cases}\tag{8.2.2}$$

由于本问题中有$p_1=p_3$，自然有$\tau_1=\tau_3$，即$P_{\mathrm{idle},1}=P_{\mathrm{idle},3}$。

对于$P_{\mathrm{s},n}$，由式(4.5.7)，注意到$\lambda_{1,2}=\lambda_{2,3}=\lambda_{1,2,3}=0,\lambda_{1,3}=1$。

代入$P_{\mathrm{e}}=0,\mathbb{D}_k,\mathbb{E}_k,\mathbb{F}_k,k\in[1,3]$，有：

$$\begin{cases}P_{\mathrm{s},1}=\tau_1(1-\tau_2)+\tau_2(1-\tau_1)(1-\tau_3)\\P_{\mathrm{s},2}=\tau_1(1-\tau_2)+\tau_2(1-\tau_1)(1-\tau_3)+\tau_3(1-\tau_2)-\tau_1(1-\tau_2)\tau_3(1-\tau_2)\\P_{\mathrm{s},3}=\tau_2(1-\tau_1)(1-\tau_3)+\tau_3(1-\tau_2)\end{cases}\tag{8.2.3}$$

代入$\tau_1=\tau_3$可见$P_{\mathrm{s},1}=P_{\mathrm{s},3}$，对于每个节点，由式(4.5.8)，可得单个节点的吞吐量为：

$$\begin{cases}S_1=\dfrac{\tau_1(1-\tau_2)\cdot E[P]}{T_{\mathrm{slot}}\cdot P_{\mathrm{idle},1}+P_{\mathrm{s},1}\cdot T_{\mathrm{s}}+\left(1-P_{\mathrm{s},1}-P_{\mathrm{idle},1}\right)T_{\mathrm{c}}}\\S_2=\dfrac{\tau_2(1-\tau_1)(1-\tau_3)\cdot E[P]}{T_{\mathrm{slot}}\cdot P_{\mathrm{idle},1}+P_{\mathrm{s},1}\cdot T_{\mathrm{s}}+\left(1-P_{\mathrm{s},1}-P_{\mathrm{idle},1}\right)T_{\mathrm{c}}}\\S_3=\dfrac{\tau_3(1-\tau_2)\cdot E[P]}{T_{\mathrm{slot}}\cdot P_{\mathrm{idle},1}+P_{\mathrm{s},1}\cdot T_{\mathrm{s}}+\left(1-P_{\mathrm{s},1}-P_{\mathrm{idle},1}\right)T_{\mathrm{c}}}\end{cases}\tag{8.2.4}$$

自然地，系统总吞吐量为$S=\sum_{n\in\mathcal{N}}S_n=S_1+S_2+S_3$。

8.2.2. 情况二：AP1 与 AP2 之间会互相干扰，AP2 与 AP3 之间不会互相干扰

此时有$\mathbb{D}_1=\{2\},\mathbb{D}_2=\{1\},\mathbb{D}_3=\emptyset,\mathbb{E}_1=\emptyset,\mathbb{E}_2=\{3\},\mathbb{E}_3=\{2\},\mathbb{F}_1=\emptyset,\mathbb{F}_2=\emptyset,\mathbb{F}_3=\emptyset$，本问题中的丢包率$P_{\mathrm{e}}=0\%$，故不考虑丢包问题。

对于每一个节点，由式(4.5.3)，代入$P_{\mathrm{e}}=0,\mathbb{D}_k,\mathbb{F}_k,k\in[1,3]$可以得到：

$$\begin{cases}p_1=1-(1-\tau_2)=\tau_2\\p_2=1-(1-\tau_1)=\tau_1\\p_3=1-1=0\end{cases}\tag{8.2.5}$$

其中$\tau_n=b_{0,0,n}[1-(p_n)^{r+1}]/(1-p_n)$，$b_{0,0,n}$见式(4.5.2)，$\tau_n$是关于$p_n$的有理多项式。

对于$P_{\mathrm{idle},n}$由式(4.5.5)，代入$P_{\mathrm{e}}=0,\mathbb{D}_k,\mathbb{E}_k,k\in[1,3]$可得：

$$\begin{cases} P_{\text{idle},1} = (1-\tau_1)(1-\tau_2) \\ P_{\text{idle},2} = (1-\tau_2)(1-\tau_1)(1-\tau_3) \\ P_{\text{idle},3} = (1-\tau_3)(1-\tau_2) \end{cases} \tag{8.2.6}$$

对于$P_{\text{s},n}$，由式(4.5.7)，注意到$\lambda_{1,3} = \lambda_{2,3} = 1$，$\lambda_{1,2} = \lambda_{1,2,3} = 0$。
代入$P_{\text{e}} = 0, \mathbb{D}_k, \mathbb{E}_k, \mathbb{F}_k, k \in [1,3]$，有：

$$\begin{cases} P_{\text{s},1} = \tau_1(1-\tau_2) + \tau_2(1-\tau_1) \\ P_{\text{s},2} = \tau_1(1-\tau_2) + \tau_2(1-\tau_1) + \tau_3 \\ \qquad -\tau_1\tau_3(1-\tau_2) - \tau_2\tau_3(1-\tau_1) + \tau_1\tau_2\tau_3(1-\tau_1)(1-\tau_2) \\ P_{\text{s},3} = \tau_2(1-\tau_1) + \tau_3 - \tau_2\tau_3(1-\tau_1) \end{cases} \tag{8.2.7}$$

对于每个节点，由式(4.5.8)，可得单个节点的吞吐量为：

$$\begin{cases} S_1 = \dfrac{\tau_1(1-\tau_2)\cdot E[P]}{T_{\text{slot}}\cdot P_{\text{idle},1} + P_{\text{s},1}\cdot T_{\text{s}} + \left(1 - P_{\text{s},1} - P_{\text{idle},1}\right)T_{\text{c}}} \\ S_2 = \dfrac{\tau_2(1-\tau_1)\cdot E[P]}{T_{\text{slot}}\cdot P_{\text{idle},1} + P_{\text{s},1}\cdot T_{\text{s}} + \left(1 - P_{\text{s},1} - P_{\text{idle},1}\right)T_{\text{c}}} \\ S_3 = \dfrac{\tau_3\cdot E[P]}{T_{\text{slot}}\cdot P_{\text{idle},1} + P_{\text{s},1}\cdot T_{\text{s}} + \left(1 - P_{\text{s},1} - P_{\text{idle},1}\right)T_{\text{c}}} \end{cases} \tag{8.2.8}$$

自然地，系统总吞吐量为$S = \sum_{n\in\mathcal{N}} S_n = S_1 + S_2 + S_3$。

8.2.3. 情况三：AP1 与 AP2 之间不会互相干扰，AP2 与 AP3 之间会互相干扰

此时有$\mathbb{D}_1 = \emptyset, \mathbb{D}_2 = \{1\}, \mathbb{D}_3 = \{2\}, \mathbb{E}_1 = \{2\}, \mathbb{E}_2 = \{3\}, \mathbb{E}_3 = \emptyset, \mathbb{F}_1 = \emptyset, \mathbb{F}_2 = \emptyset, \mathbb{F}_3 = \emptyset$，本问题中的丢包率$P_{\text{e}} = 0\%$，故不考虑丢包问题。
对于每一个节点，由式(4.5.3)，代入$P_{\text{e}} = 0, \mathbb{D}_k, \mathbb{F}_k, k \in [1,3]$可以得到：

$$\begin{cases} p_1 = 1 - 1 = 0 \\ p_2 = 1 - (1-\tau_1) = \tau_1 \\ p_3 = 1 - (1-\tau_2) = \tau_2 \end{cases} \tag{8.2.9}$$

其中$\tau_n = b_{0,0,n}[1 - (p_n)^{r+1}]/(1-p_n)$，$b_{0,0,n}$见式(4.5.2)，$\tau_n$是关于$p_n$的有理多项式。
对于$P_{\text{idle},n}$，由式(4.5.5)，代入$P_{\text{e}} = 0, \mathbb{D}_k, \mathbb{E}_k, k \in [1,3]$可得：

$$\begin{cases} P_{\text{idle},1} = (1-\tau_1)(1-\tau_2) \\ P_{\text{idle},2} = (1-\tau_2)(1-\tau_1)(1-\tau_3) \\ P_{\text{idle},3} = (1-\tau_3)(1-\tau_2) \end{cases} \tag{8.2.10}$$

对于$P_{\text{s},n}$，由式(4.5.7)，注意到$\lambda_{1,3} = \lambda_{1,2} = 1$，$\lambda_{2,3} = \lambda_{1,2,3} = 0$。
代入$P_{\text{e}} = 0, \mathbb{D}_k, \mathbb{E}_k, \mathbb{F}_k, k \in [1,3]$，有：

$$\begin{cases} P_{\text{s},1} = \tau_1 + \tau_2(1-\tau_3) - \tau_1\tau_2(1-\tau_3) \\ P_{\text{s},2} = \tau_1 + \tau_2(1-\tau_3) + \tau_3(1-\tau_2) \\ \qquad -\tau_1\tau_2(1-\tau_3) - \tau_1\tau_3(1-\tau_2) + \tau_1\tau_2\tau_3(1-\tau_2)(1-\tau_3) \\ P_{\text{s},3} = \tau_2(1-\tau_3) + \tau_3(1-\tau_2) \end{cases} \tag{8.2.11}$$

对于每个节点，由式(4.5.8)，可得单个节点的吞吐量为：

$$\begin{cases} S_1 = \dfrac{\tau_1 \cdot E[P]}{T_{\text{slot}} \cdot P_{\text{idle},1} + P_{\text{s},1} \cdot T_{\text{s}} + \left(1 - P_{\text{s},1} - P_{\text{idle},1}\right) T_{\text{c}}} \\ S_2 = \dfrac{\tau_2 (1 - \tau_1) \cdot E[P]}{T_{\text{slot}} \cdot P_{\text{idle},1} + P_{\text{s},1} \cdot T_{\text{s}} + \left(1 - P_{\text{s},1} - P_{\text{idle},1}\right) T_{\text{c}}} \\ S_3 = \dfrac{\tau_3 (1 - \tau_2) \cdot E[P]}{T_{\text{slot}} \cdot P_{\text{idle},1} + P_{\text{s},1} \cdot T_{\text{s}} + \left(1 - P_{\text{s},1} - P_{\text{idle},1}\right) T_{\text{c}}} \end{cases} \tag{8.2.12}$$

自然地，系统总吞吐量为$S = \sum_{n \in \mathcal{N}} S_n = S_1 + S_2 + S_3$。

8.2.4. 情况四：AP1 与 AP2 之间不会互相干扰，AP2 与 AP3 之间不会互相干扰

此时有$\mathbb{D}_1 = \emptyset, \mathbb{D}_2 = \emptyset, \mathbb{D}_3 = \emptyset, \mathbb{E}_1 = \{2\}, \mathbb{E}_2 = \{1,3\}, \mathbb{E}_3 = \{2\}, \mathbb{F}_1 = \emptyset, \mathbb{F}_2 = \emptyset, \mathbb{F}_3 = \emptyset$，本问题中的丢包率$P_{\text{e}} = 0\%$，故不考虑丢包问题。

对于每一个节点，由式(4.5.3)，代入$P_{\text{e}} = 0, \mathbb{D}_k, \mathbb{F}_k, k \in [1,3]$可以得到：

$$\begin{cases} p_1 = 1 - 1 = 0 \\ p_2 = 1 - 1 = 0 \\ p_3 = 1 - 1 = 0 \end{cases} \tag{8.2.13}$$

其中$\tau_n = b_{0,0,n}[1 - (p_n)^{r+1}]/(1 - p_n)$，$b_{0,0,n}$见式(4.5.2)，$\tau_n$是关于$p_n$的有理多项式。

对于$P_{\text{idle},n}$，由式(4.5.5)，代入$P_{\text{e}} = 0, \mathbb{D}_k, \mathbb{E}_k, k \in [1,3]$可得：

$$\begin{cases} P_{\text{idle},1} = (1 - \tau_1)(1 - \tau_2) \\ P_{\text{idle},2} = (1 - \tau_2)(1 - \tau_1)(1 - \tau_3) \\ P_{\text{idle},3} = (1 - \tau_3)(1 - \tau_2) \end{cases} \tag{8.2.14}$$

对于$P_{\text{s},n}$，由式(4.5.7)，注意到$\lambda_{1,3} = \lambda_{1,2} = \lambda_{2,3} = \lambda_{1,2,3} = 1$。

代入$P_{\text{e}} = 0, \mathbb{D}_k, \mathbb{E}_k, \mathbb{F}_k, k \in [1,3]$，有：

$$\begin{cases} P_{\text{s},1} = \tau_1 + \tau_2 - \tau_1\tau_2 \\ P_{\text{s},2} = \tau_1 + \tau_2 + \tau_3 - \tau_1\tau_2 - \tau_1\tau_3 - \tau_2\tau_3 + \tau_1\tau_2\tau_3 \\ P_{\text{s},3} = \tau_3 + \tau_2 - \tau_2\tau_3 \end{cases} \tag{8.2.15}$$

对于每个节点，由式(4.5.8)，可得单个节点的吞吐量为：

$$\begin{cases} S_1 = \dfrac{\tau_1 \cdot E[P]}{T_{\text{slot}} \cdot P_{\text{idle},1} + P_{\text{s},1} \cdot T_{\text{s}} + \left(1 - P_{\text{s},1} - P_{\text{idle},1}\right) T_{\text{c}}} \\ S_2 = \dfrac{\tau_2 \cdot E[P]}{T_{\text{slot}} \cdot P_{\text{idle},1} + P_{\text{s},1} \cdot T_{\text{s}} + \left(1 - P_{\text{s},1} - P_{\text{idle},1}\right) T_{\text{c}}} \\ S_3 = \dfrac{\tau_3 \cdot E[P]}{T_{\text{slot}} \cdot P_{\text{idle},1} + P_{\text{s},1} \cdot T_{\text{s}} + \left(1 - P_{\text{s},1} - P_{\text{idle},1}\right) T_{\text{c}}} \end{cases} \tag{8.2.16}$$

自然地，系统总吞吐量为$S = \sum_{n \in \mathcal{N}} S_n = S_1 + S_2 + S_3$。

8.3. 模型理论求解

对于情况一，首先将式(4.5.1)、式(8.2.1)和联立，得到只含有 AP1，AP2，AP3 的碰撞概率 p_1，p_2，p_3的多项式方程，求解出 p_1，p_2，p_3的数值解，进而得到节点在某个时隙发送数据帧的概率 τ_1，τ_2，τ_3，以及马尔可夫链中$b_{0,0,1}$，$b_{0,0,2}$，$b_{0,0,3}$ 的值。最后根据题目中的参数要求，由式(8.2.2)，式(8.2.3)和式(8.2.4)计算不同吞吐量，不同最小窗口和最大窗口，最大重传次数场景下的理论吞吐量。其中，物理层速率，最小窗口和最大窗口及最大重传次数的可选范围如表 8-1 所示。

$W_{\min}$	$W_{\max}$	最大重传次数	物理层速率
16	1024	6	455.8Mbps
32	1024	5	455.8Mbps
16	1024	32	455.8Mbps
16	1024	6	286.8Mbps
32	1024	5	286.8Mbps
16	1024	32	286.8Mbps
16	1024	6	158.4Mbps
32	1024	5	158.4Mbps
16	1024	32	158.4Mbps

表 8-1 最小窗口和最大窗口，最大重传次数的取值范围

通过 MATLAB 数值计算，最终得到在不同物理层速率，最小窗口和最大窗口，最大重传次数的条件下，系统的吞吐量性能和冲突概率如图 8-1 和图 8-2 所示。

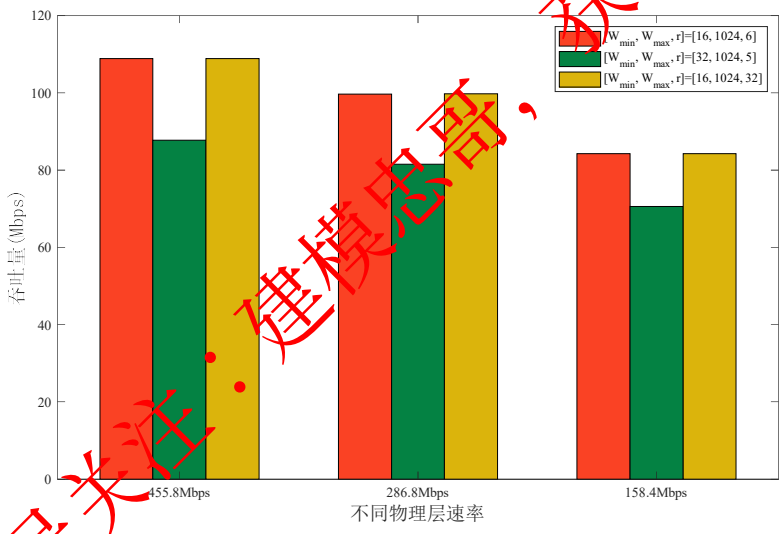

图 8-1 不同物理层速率下理论系统吞吐量

由图 8-1 可知，在 3 BSS 系统场景下，物理层速率增大，系统成功发送单个数据包的期望时间减小，系统吞吐量增加。对比红色柱与绿色柱可以得到，最小窗口的大小对系统吞吐量有一定程度影响，最小窗口增大，导致退避窗口增大，提高了节点处于退避时间的期望，最终吞吐量减小。同时，比较红色柱和黄色柱，最大重传次数增加对当前系统的影响较小，这可能是系统中节点数量较少，发生连续碰撞的可能性较低，难以达到最大重传次数，因此增大最大重传次数对系统吞吐量影响不明显。

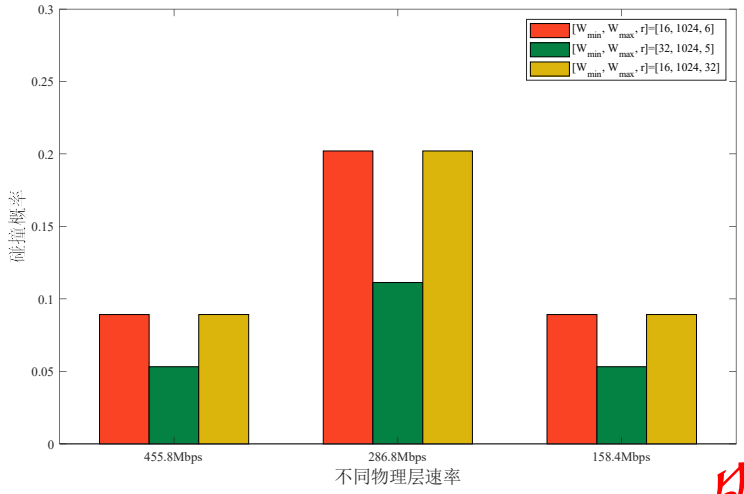

图 8-2 不同物理层速率下理论系统碰撞概率

从图 8-2 可知，物理层速率对系统碰撞概率并无影响，455.8Mbps 与 158.4Mbps 物理层速率下，系统碰撞概率相同，而 286.8Mbps 物理层速率下系统碰撞概率是 455.8Mbps 物理层速率的两倍。然而在相同物理层速率下，如图 8-2 中红色柱与绿色柱对比，增加最小窗口大小，退避窗口大小增加，使得节点退避时间的期望值增加，系统碰撞概率降低。此外，对比红色柱与黄色柱，最大重传次数对碰撞概率的影响不明显，这可能是因为网络中节点较少，发生多次碰撞的可能性较低，很难达到最大重传次数，因此最大重传次数对系统碰撞概率的影响不明显。

由于此场景中不同物理层速率对系统碰撞概率没有影响，并且在 286.8Mbps 物理层速率下系统的碰撞概率是其他速率的两倍，考虑到启用 RTS/CTS 机制在系统碰撞概率较大时可以提升系统的吞吐量，因此启用 RTS/CTS 机制进行仿真验证，此时系统吞吐量如图 8-3 所示。

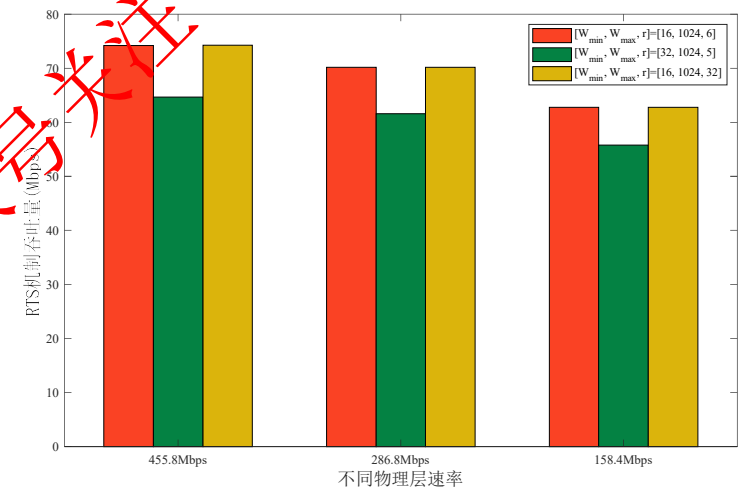

图 8-3 不同物理层速率下启用 RTS/CTS 时的系统吞吐量

从图 8-3 中可以得知，在此场景下，启用 RTS/CTS 并不能提升系统的吞吐量。这是因为在 3 BSS 场景下，节点发生碰撞的概率并不如隐蔽站场景下高，启用 RTS/CTS 时减小碰撞的收益低于发送 RTS/CTS 帧所增加的时间开销，因此在此场景下不推荐使用 RTS/CTS

机制。

对于情况二和情况三，系统的节点关系是对称的，因此系统的吞吐量应该是一致的，通过仿真得到这两种情况下的系统吞吐量曲线如图 8-4 所示：

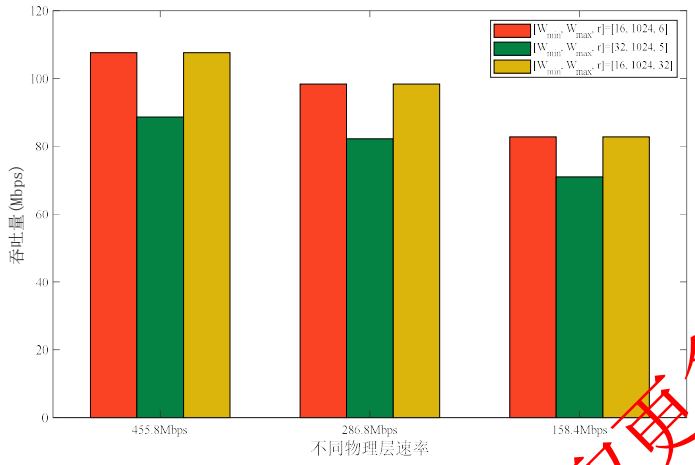

图 8-4 情况二和情况三不同场景下的系统吞吐量

从图 8-4 可以得知，在情况二和情况三中，系统的吞吐量与情况一基本接近。

对于情况四，通过仿真得到这种情况下的系统吞吐量曲线如图 8-5 所示，从图中可以看出此种情况下系统吞吐量略高于前三种场景，可能的原因是节点发包时无干扰，没有因碰撞导致丢包的情况。

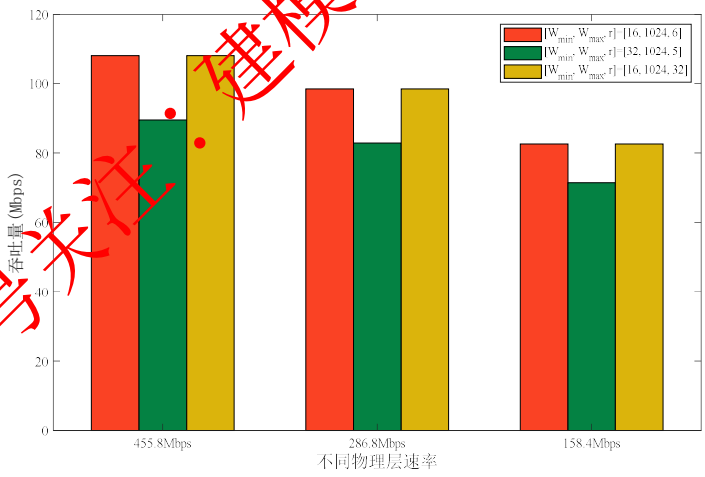

图 8-5 情况四不同场景下的系统吞吐量

8.4. 模型仿真求解

8.4.1. 物理层速率为 455.8Mbps 下场景

首先通过本文 MATLAB 仿真器，对$B = 455.8\text{Mbps}, W_{\min} = 16, W_{\max} = 1024, r = 32$的场景进行了仿真验证，得到系统总吞吐量与碰撞概率如表 8-2 所示，对情况一进行了仿真。得到仿真事件图 8-6，系统吞吐量曲线图 8-7。

系统总吞吐量	节点碰撞概率
103.086Mbps	14.8%

表 8-2 B = 455.8Mbps时问题四系统吞吐及碰撞概率

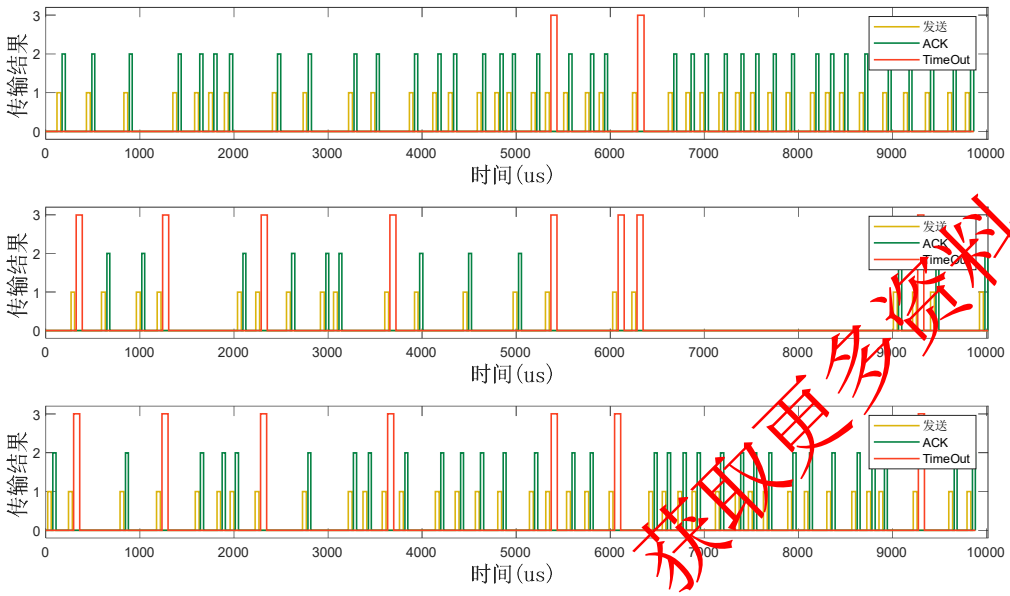

图 8-6 B = 455.8Mbps时问题四仿真事件图

图 8-7 为物理层速率为 455.8Mbps 情况下三个 AP 的仿真事件图，从上至下依次是，AP1、AP2 和 AP3，其他参数与表 5-1 一致。图 8-7 为 3 BSS 系统的吞吐量变化曲线，其中红色代表系统总吞吐量，黄色代表 AP3 吞吐量，橘红色代表 AP1 吞吐量，绿色代表 AP2 吞吐量。由问题分析可知，AP2 同时监听 AP1 与 AP3，因此 AP2 大部分时间处于等待状态，但是等待时间和退避时间过去后，又会在发送时频繁地与 AP1 或 AP3 冲突，与图 8-6 呈现的仿真事件保持一致，自然地，AP2 的吞吐量是最低的，在图 8-7 中代表 AP2 吞吐量的绿色线条处于最底部。AP1 与 AP3 是对称关系，AP1 与 AP3 能够并行传输并且互不影响，在竞争信道的能力上处于优势地位，在相同的时间段内绝大多数时间都在发送数据，因此 AP1 与 AP3 的吞吐量是 AP2 的两倍，如图 8-7 所示。

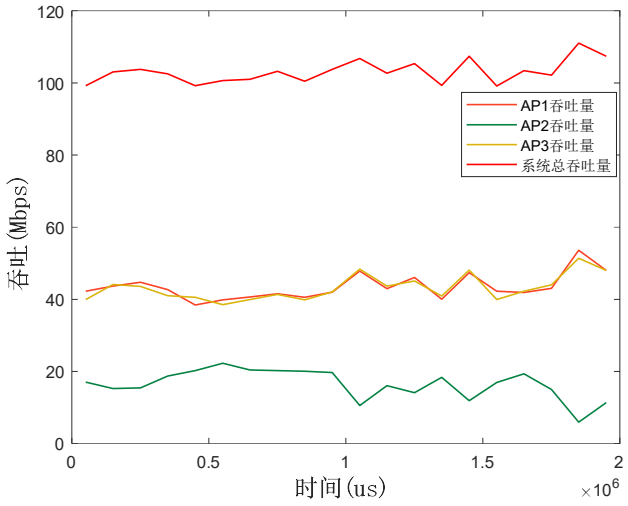

图 8-7 B = 455.8Mbps时问题四系统的吞吐量

8.4.2. 物理层速率为 286.8Mbps 和 158.4Mbps 情况

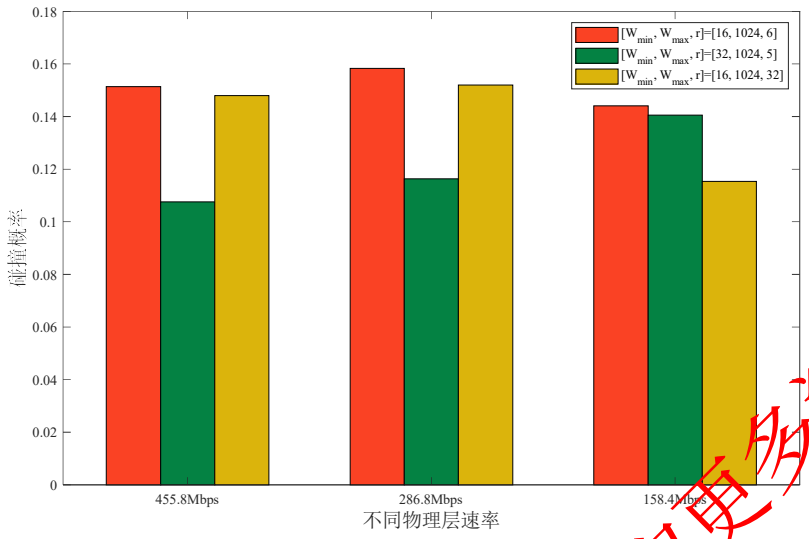

图 8-8 问题四不同物理层速率下系统碰撞概率

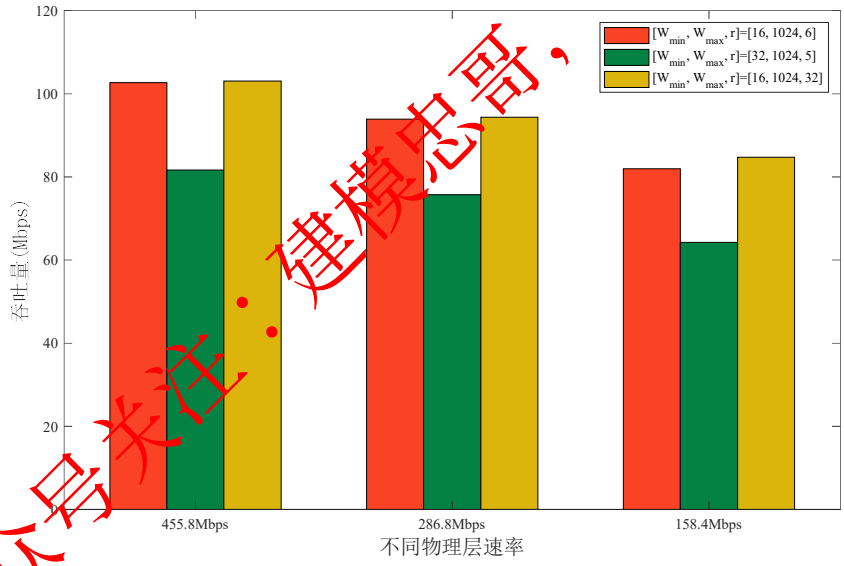

图 8-9 问题四不同物理层速率下系统吞吐量

由图 8-8 柱状图可知，从纵向比较来看，不同物理层速率影响系统的碰撞概率，物理层速度越大，系统碰撞概率越小。从横向来看，在相同的物理层速率之下，由于 3 BSS 系统的竞争比较激烈，最小竞争窗口越大，节点退避时间的期望越大，再次发送数据冲突的概率越低，因此系统的碰撞概率越小，符合实验预期。而最大重传次数对物理层速率较小的场景作用更明显，如图 8-8 中 158.4Mbps 场景下的黄色柱，因为物理层速率越小，系统碰撞的概率越大，而增加最大重传次数能够明显增加退避时间的期望，降低节点在冲突后再次冲突的概率，达到减小冲突的目的，降低了系统的碰撞概率。

图 8-9 柱状图描述了在不同物理层速率下 3 BSS 系统，改变竞争窗口和最大重传次数对吞吐量的影响，纵向对比来看，物理层速率越大，系统成功发送单个数据包的期望时间减小，系统的吞吐量越大。从横向比较，对比红色柱与绿色柱，当最小竞争窗口增加，节

点退避时间的期望就会增加，这会降低系统的吞吐量。对比红色柱与黄色柱，当最大重传次数增加，系统总吞吐量的变化并不明显，这是因为 AP2 长时间处于等待和退避状态，AP1 与 AP3 能够并行传输，因此系统中的发生多次碰撞的概率较低，增大最大重传次数对系统总吞吐量变化并不明显。

物理层速率	$W_{\min}$	最大重传次数	理论吞吐量(Mbps)	系统吞吐量(Mbps)	误差
455.8Mbps	16	6	108.8642	102.708	5.655%
	32	5	87.7287	81.654	6.924%
	16	32	108.8863	103.086	5.327%
286.8Mbps	16	6	99.7097	93.918	5.809%
	32	5	81.5024	75.696	7.124%
	16	32	99.7302	94.398	5.347%
158.4Mbps	16	6	84.2254	81.984	2.661%
	32	5	70.5549	64.224	8.973%
	16	32	84.2432	84.744	0.594%

表 8-3 3BSS 系统在不同场景下的吞吐量和碰撞概率

根据问题四的要求，对情况一进行了仿真。本文在仿真器中仿真了 3 BSS 系统在 3 种不同物理层速率下，改变最小竞争窗口和最大重传次数得到不同状态下系统的吞吐量和碰撞概率，共 9 组数据如上表所示：根据表格可知，理论吞吐量和仿真吞吐量的平均误差在 5%左右，证实了理论模型的正确性和仿真系统的有效性。

9. 模型评价

9.1. 模型的优点

1. 本文在 Bianchi 模型的基础上，考虑了信道接入具有最大重传次数的场景，将原模型对应的无穷级数求和问题转换为有限长数列求和问题。
2. 本文对已有的模型进行了综合与推广，得到了同时考虑隐藏终端，暴露终端和自然丢包等复杂网络场景下的统一网络整体吞吐模型。
3. 通过分析网络中每个节点的互听节点和干扰节点，得到了统一的单节点的吞吐模型。该模型可以分析非对称复杂网络场景下的系统理论吞吐。
4. 分析了 RTS/CTS 机制对不同网络场景下吞吐性能的影响。证明了 RTS/CTS 机制在冲突概率较高的隐藏终端场景可以提升网络吞吐。在暴露终端场景会降低网络吞吐。
5. 本文基于 MATLAB 编写了一个离散事件仿真器，可以模拟不同网络场景中每个节点的退避、等待、传输、冲突和超时重传等网络状态，可以对系统中每个节点在任意时刻的状态进行可视化展示。

9.2. 模型的缺点

1. 当前的离散事件仿真器不支持模拟具有 RTS/CTS 机制的场景，应考虑加入此通信机制，比较在不同场景下采用 RTS/CTS 机制对系统吞吐性能的影响。
2. 使用 MATLAB 仿真有着较大的局限性，仿真速度较慢，未来应考虑使用更专业的仿真器，如 ns-3 进行系统吞吐仿真验证。

9.3. 未来的展望

无线局域网在我们的日常生活中扮演着重要的作用，而信道接入机制对网络的整体吞吐有重要影响。在实际的应用场景中，不同的物理层速率，包有效载荷的长度，退避机制都会对信道接入的效率有重要影响。

在复杂的网络场景中，单一的接入方式往往不能够得到良好的网络吞吐性能。信道接入的理论模型往往假设网络中移动站以相同速率传输数据。然而，在如今的无线局域网中，移动站常常根据瞬时信道状态调整数据速率，因此多种数据速率可以在网络中共存。未来应考虑为多数据速率网络场景，并分析 RTS/CTS 机制对系统吞吐的影响。

由于时间有限，本题使用的基于 MATLAB 编写的仿真器只有部分系统参数可选，未来可以使用更专业的仿真器，如 ns-3，在已有的研究[4]上加入我们的研究工作，进一步拓展模型对复杂系统场景的适应性。

参考文献

[1] Yeh J H, Chen J C, Lee C C. WLAN standards[J]. IEEE Potentials, 2003, 22(4): 16-22.

[2] Bianchi G. IEEE 802.11-saturation throughput analysis[J]. IEEE communications letters, 1998, 2(12): 318-320.

[3] Hung F Y, Marsic I. Performance analysis of the IEEE 802.11 DCF in the presence of the hidden stations[J]. Computer Networks, 2010, 54(15): 2674-2687.

[4] Patidar R, Roy S, Henderson T R, et al. Validation of Wi-Fi network simulation on ns-3[J]. University of Washington Technical Report, 2017.

附录 A 仿真器程序与运行说明

A.1. 程序结构说明

本仿真程序是基于 MATLAB R2021a 进行开发的，基于 MATLAB 的面向对象编程工具，将每个 AP 封装为了一个类，并采用元组传递拓扑关系。所有 AP 通过一个 Simulator 类进行管理与交互，AP 将自己的行为抽象为事件，Simulator 将循环遍历所有 AP 的事件，并按照时间顺序从前向后执行。

本程序只需改动 main.m 文件的参数配置，即可对不同场景进行仿真。

A.2. 主脚本的代码与运行说明

主脚本为 main.m 函数，问题一的运行示例如下代码所示。

每个 AP 类通过结构体 simu_conf 进行构造，其中 mbps, simulator, p_e 是必选参数，W_min, W_max 是可选参数。

构造完 AP 后，将它们组成一个元组，同时传入参数 `topoAP2AP`，`topoAP2STA`，这两个参数定义了网络的拓扑关系，该代码对应了问题一的场景。

`simulator.Init` 初始化仿真，`simulator.RunAll` 执行仿真，`simulator.PlotAllEvent` 将绘制事件的时序图，`simulator.PlotThroughput` 将统计每个 AP 的吞吐量、碰撞概率。

详细代码如下所示：

```
clear AP;
clear Callback;
close all;clear;clc;
rng(0);                    % 固定随机数种子，便于复现
% 1. 参数配置
simu_conf = struct();
max_time = 2e6;             % 最大仿真时间 (us)
simulator = Simulator(max_time);
simu_conf.mbps = 455.8;     % 物理层速率
% simu_conf.mbps = 275.3;     % 物理层速率
% simu_conf.mbps = 158.4;     % 物理层速率
simu_conf.simulator = simulator;
simu_conf.r = 6;            % 最大重传次数
simu_conf.p_e = 0;          % 信道质量导致的丢包率
simu_conf.W_min = 16;
% 2. 创建 AP
APs = cell(2, 1);
APs{1} = AP(simu_conf);     % AP1
APs{2} = AP(simu_conf);     % AP2
% 3. 创建拓扑
% topoAP2AP 代表每个 AP 会影响到的别的 AP 的 id 号
topoAP2AP = {{2}, {1}};
% topoAP2STA 代表每个 AP 会影响到的别的 STA 的 id 号
topoAP2STA = {{2}, {1}};
% 4. 启动仿真
```

```
simulator.Init(APs, topoAP2AP, topoAP2STA);
simulator.RunAll();
% 5. 绘制仿真输出
simulator.PlotAllEvent();
% 6.绘制吞吐曲线
% avg_throughput 是平均吞吐量，p 是碰撞概率
[avg_throughput, p] = simulator.PlotThroughput(1e4);
```

A.3. 全部代码

由于本程序具有着良好的封装，原则上是不需要修改其余文件的代码的。此处将其余文件的代码汇总于此。

1. ApState.m

```
classdef ApState < uint32
    enumeration
        BACK_OFF (1);   % 正在 backoff
        HOLD     (2);   % 正在 hold
        SEND     (3);   % 正在发送 (t_DATA + t_SIFS + t_ACK)
                        % 等待 TimeOut ((t_TO - t_ACK - ap.t_SIFS) + t_DIFS)
                        % 等待 DIFS (t_DIFS)
    end
end
```

2. Callback.m

```
classdef Callback < handle
    properties
        t;          % 执行时间
        lambda;     % 执行函数
        uid;        % UID
    end

    methods
        function obj = Callback(t, lambda)
            obj.t = t;
            obj.lambda = lambda;
            persistent uid0;
            if isempty(uid0)
                uid0 = 1;
            end
            obj.uid = uid0;
            uid0 = uid0 + 1;
        end
    end
end
```

3. EventState.m

```
classdef EventState < uint32
```

```
    enumeration
        SEND (1);   % 发送事件
        ACK  (2);   % 发送成功
        TO   (3);   % 发送失败
    end
end
```

4. Events.m

```
classdef Events < handle
    properties
        t_start;
        t_end;
        state;
    end
    methods
        function obj = Events(t_start, t_end, state)
            obj.t_start = t_start;
            obj.t_end = t_end;
            obj.state = state;
        end
    end
end
```

5. Simulator.m

```
classdef Simulator < handle
    properties
        t;
        max_time;   % 最大仿真时长(us)
        APs;        % AP 组成的 cell
        topoAP2AP;  % topoAP2AP 代表每个 AP 会影响到的别的 AP 的 id 号
        topoAP2STA; % topoAP2STA 代表每个 AP 会影响到的别的 STA 的 id 号
    end
    methods
        % 1.空构造
        function obj = Simulator(max_time)
            obj.max_time = max_time;
            % 请调用 init 实现初始化
        end
        % 2. 构造函数的实现，以达成延迟构造
        function Init(obj, APs, topoAP2AP, topoAP2STA)
            obj.t = 0;
            obj.APs = APs;
            obj.topoAP2AP = topoAP2AP;
            obj.topoAP2STA = topoAP2STA;

            for i = 1:length(obj.APs)
                obj.APs{i}.effect_AP = topoAP2AP{i};
```

```
                obj.APs{i}.effect_STA = topoAP2STA{i};
                obj.APs{i}.Run();
            end
        end
        % 3. 主循环
        function RunAll(obj)
            while true
                obj.Run();
                if obj.t > obj.max_time
                    return;
                end
            end
        end
        % 4. 运行时间最早的回调函数
        function Run(obj)
            min_idx = 0;
            min_t = 1e18;
            for i = 1:length(obj.APs)
                ap = obj.APs{i};
                if ~isempty(ap.callback) ...
                    && (ap.callback.t < min_t || ...
                        (ap.callback.t == min_t && ap.callback.uid <
obj.APs{min_idx}.callback.uid))
                    min_idx = i;
                    min_t = ap.callback.t;
                end
            end
            assert(min_idx ~= 0);
            obj.t = min_t;
            lambda = obj.APs{min_idx}.callback.lambda;
            obj.APs{min_idx}.callback = [];
            lambda();
        end
        % 5. 绘制事件曲线
        function PlotAllEvent(obj)
            figure('Name', '仿真事件曲线');
            cc = {'#DBB40C', '#048243','#FA4224'};
            for i = 1:length(obj.APs)
                events = obj.APs{i}.all_event;
                subplot(length(obj.APs), 1, i);
                for k = 1:3
                    x = (0);
                    y = (0);
                    for j = 1:length(events)
```

```
                if(events{j}.state == k)
                    x(end + 1) = events{j}.t_start;
                    y(end + 1) = events{j}.state;
                    x(end + 1) = events{j}.t_end;
                    y(end + 1) = 0;
                end
            end
            x(end + 1) = events{length(events)}.t_end + 1;
            y(end + 1) = 0;
            s = stairs(x, y, 'Color', cc{k}, 'LineWidth', 1);
            hold on;
        end
        hold off;
        legend('发送','ACK','TimeOut');
        %修改 x,y 轴标签
        ylabel('\fontname{宋体}\fontsize{14}传输结果');
        xlabel('\fontname{宋体}\fontsize{14}时间');
        axis([0, obj.t + 1, -0.2, 3.2])
    end
end
% 6. 绘制吞吐量曲线
function [avg_throughput, p_c] = PlotThroughput(obj, gap)
    figure('Name', '吞吐量曲线');
    count_ack = zeros(floor(obj.max_time / gap) + 1, 1);
    count_to = zeros(floor(obj.max_time / gap) + 1, 1);
    for i = 1:length(obj.APs)
        events = obj.APs{i}.all_event;
        for j = 1:length(events)
            now_t = events{j}.t_start;
            idx = floor(now_t / gap) + 1;
            if events{j}.state == EventState.ACK
                count_ack(idx) = count_ack(idx) + 1;
            elseif events{j}.state == EventState.TO
                count_to(idx) = count_to(idx) + 1;
            end
        end
    end
    count_ack = count_ack(1:end-1);
    count_to = count_to(1:end-1);
    throughput = count_ack * 1500 * 8 / gap;
    x = (gap/2) : gap : (obj.max_time + gap/2);
    x = x(1:length(count_ack));
    plot(x, throughput, 'r');
    ax = gca;
```

```
            ax.YLim = [0, ceil(max(throughput) / 10) * 10];
            xlabel('时间(us)');
            ylabel('吞吐(Mbps)');
            avg_throughput = mean(throughput);
            p_c = (sum(count_to)) / (sum(count_ack) + (sum(count_to)));
        end
    end
end
```

6. AP.m

```
% 包含一个 AP 的属性和状态
classdef AP < handle
    % 1. 数据成员（原则上非友元类仅可访问，不可随意修改）
    properties (SetAccess = public, GetAccess = public)
        last_t;         % 上一次事件的时间
        ords;           % 发送次数
        t_window;       % 当前窗口的剩余时间 (us)
        state;          % 状态，取值在 ApState 枚举类型中
        simulator;      % 仿真器
        callback;       % 回调定时
        all_event;      % 全部事件追踪
        sta_event;      % 自己 STA 被干扰的事件
        effect_AP;      % 能听到自己发包的 AP
        effect_STA;     % 能被自己发包干扰的 STA
        mbps;           % 带宽 (Mbps)
        r;              % 最大重传次数
        dt;             % t_PHY + (B_MAC + B_Payload) * 8 / 带宽 (us)
        W_min;          % CW_min
        W_max;          % CW_max
        p_e;            % 信道质量导致的丢包率
        id;             % UID，标识 AP 号
    end
    % 2. 类的通用属性
    properties (Constant)
        t_ACK         = 32;     % ACK (us)
        t_SIFS        = 16;     % SIFS (us)
        t_DIFS        = 43;     % DIFS (us)
        t_SLOT        = 9;      % SLOT (us)
        t_TO          = 65;     % ACK TimeOut (us)
        t_PHY         = 13.6;   % PHY 头时长 (us)
        L_MAC         = 30;     % MAC 头 (Byte)
        L_PLD         = 1500;   % Payload (Byte)
        default_W_min = 16;     % 默认的 CW_min
        default_W_max = 1024;   % 默认的 CW_max
    end
```

```
    % 3. 公有接口方法
    methods (Access = public)
        % 1. 构造函数，接受必选参数 {mbps, simulator, r, p_e} 和可选参数 {W_min,
W_max} 对 AP 进行配置
        function obj = AP(simu_conf)
            % simu_conf 为结构体，包含配置参数，将按照配置参数设置 AP 属性
            % 1. mbps
            if isfield(simu_conf, 'mbps')
                obj.mbps = simu_conf.mbps;
                obj.dt = obj.t_PHY + (obj.L_MAC + obj.L_PLD) * 8 / obj.mbps;
            else
                error('错误：mbps 是 AP 的必选参数，但是传入的参数没有这一条！')
            end
            % 2. W_min
            if isfield(simu_conf, 'W_min')
                obj.W_min = simu_conf.W_min;
            else
                obj.W_min = obj.default_W_min;
            end
            % 3. W_max
            if isfield(simu_conf, 'W_max')
                obj.W_max = simu_conf.W_max;
            else
                obj.W_max = obj.default_W_max;
            end
            % 4. r
            if isfield(simu_conf, 'r')
                obj.r = simu_conf.r;
            else
                error('错误：r 是 AP 的必选参数，但是传入的参数没有这一条！')
            end
            % 5. r
            if isfield(simu_conf, 'p_e')
                obj.p_e = simu_conf.p_e;
            else
                error('错误：p_e 是 AP 的必选参数，但是传入的参数没有这一条！')
            end
            % 6. id 自动编号
            persistent id0;
            if isempty(id0)
                id0 = 1;
            end
            obj.id = id0;
            id0 = id0 + 1;
```

```
        % 7. simulator
        if isfield(simu_conf, 'simulator')
            obj.simulator = simu_conf.simulator;
        else
            error('错误：simulator 是 AP 的必选参数，但是传入的参数没有这一条！')
        end
        % 8. 初始化滑动窗口与状态
        obj.last_t = 0;
        obj.ords = 0;
        obj.t_window = obj.GetWindowTime();
        obj.state = ApState.BACK_OFF;
        obj.callback = [];
    end
    % 2. 获取窗口等待时间
    function t = GetWindowTime(obj)
        if obj.W_min * 2^obj.ords > obj.W_max
            window = obj.W_max;
        else
            window = obj.W_min * 2^obj.ords;
        end
        t = randi([0, window - 1], 1, 1) * obj.t_SLOT;
    end
    % 3. 执行窗口倒计时
    function Run(obj)
        % 经过的时间差
        t = obj.simulator.t;
        delta_t = t - obj.last_t;
        obj.last_t = t;
        assert(obj.state == ApState.BACK_OFF);
        assert(isempty(obj.callback));
        % 如果当前有干扰，说明应该执行 hold
        obj.callback = Callback(t + obj.t_window, ...
                                @()obj.Send());
    end
    % 4. 尝试发送
    function Send(obj)
        % 经过的时间差
        t = obj.simulator.t;
        delta_t = t - obj.last_t;
        obj.last_t = t;
        assert(obj.state == ApState.BACK_OFF);
        % 1e-6 用于容许浮点数计算精度问题
        assert(delta_t <= obj.t_window + 1e-6);
        obj.t_window = max(0, obj.t_window - delta_t);
```

```
        % 如果应该发送
        assert(abs(obj.t_window) < 1e-6)
        obj.t_window = 0;
        obj.state = ApState.SEND;
        % 让互听的 AP 进行 hold;
        APs = obj.simulator.APs;
        for i = 1:length(obj.effect_AP)
            ap = APs{obj.effect_AP{i}};
            ap.Hold();
        end
        % 让干扰的 STA 设置干扰事件;
        for i = 1:length(obj.effect_STA)
            ap = APs{obj.effect_STA{i}};
            ap.sta_event{end + 1} = Events(t, t + obj.dt, EventState.SEND);
        end
        % 记录发送事件
        obj.all_event{end + 1} = Events(t, t + obj.dt, EventState.SEND);
        % 等到 ACK 时检查干扰
        obj.callback = Callback(t + obj.dt + obj.t_SIFS + obj.t_ACK, ...
                                @()obj.CheckAck(t, t + obj.dt));
    end
    % 5. 被别的节点设置为 hold 状态
    function Hold(obj)
        % 经过的时间差
        t = obj.simulator.t;
        delta_t = t - obj.last_t;
        obj.last_t = t;
        % 1. Back off
        if obj.state == ApState.BACK_OFF
            % 1e-6 用于容许浮点数计算精度问题
            assert(delta_t <= obj.t_window + 1e-6);

            obj.t_window = max(0, obj.t_window - delta_t);
            if obj.t_window < obj.t_SLOT - 1e-6
                % 如果在最后一个时隙内，我们认为此时 CCA 无法这么快响应
                % 那么该节点将拒绝 hold，从而导致冲突发生
            else
                % 先 hold 到 ACK，期间删除 callback，由别人重新建立
                obj.state = ApState.HOLD;
                obj.callback = [];
            end
        elseif obj.state == ApState.HOLD
            % hold 状态不用再 hold
        elseif obj.state == ApState.SEND
```

```
            % 发送过程中不用退避
        else
            error('逻辑错误：不期望的状态 UNKNOWN？ ')
        end
    end
    % 6. 解除 hold 状态
    function Unhold(obj)
        % 经过的时间差
        t = obj.simulator.t;
        delta_t = t - obj.last_t;
        obj.last_t = t;
        if obj.state == ApState.BACK_OFF
            % 1e-6 用于容许浮点数计算精度问题
            assert(delta_t <= obj.t_window + 1e-6);

            obj.t_window = max(0, obj.t_window - delta_t);
        elseif obj.state == ApState.HOLD
            obj.state = ApState.BACK_OFF;
            obj.callback = Callback(t + obj.t_window, ...
                                    @()obj.Send());
        elseif obj.state == ApState.SEND
            % 发送过程中不用取消退避
        else
            error('逻辑错误：不期望的状态 UNKNOWN？ ')
        end
    end
    % 7. 检查 ACK
    function CheckAck(obj, t1, t2)
        % 经过的时间差
        t = obj.simulator.t;
        delta_t = t - obj.last_t;
        obj.last_t = t;
        assert(obj.state == ApState.SEND);
        % 清理事件以免膨胀
        clear_idx = [];
        if length(obj.sta_event) >= 32
            for i = 1:length(obj.sta_event)
                if obj.sta_event{i}.t_end < t1
                    clear_idx(end + 1) = i;
                end
            end
            obj.sta_event(clear_idx) = [];
        end
        % 检查是否存在重叠事件
```

```
            is_ACK = true;
            for i=1:length(obj.sta_event)
                if (t1 <= obj.sta_event{i}.t_start && obj.sta_event{i}.t_start <=
t2) || (t1 <= obj.sta_event{i}.t_end && obj.sta_event{i}.t_end <= t2)
                    is_ACK = false;
                    break;
                end
            end
            % 即使收到 ACK，由于信道质量，也有 p_e 概率丢包
            rnd = rand(1, 1);
            if rnd < obj.p_e
                is_ACK = false;
            end
            if is_ACK
                % 记录收到 ACK
                obj.all_event{end + 1} = Events(t - obj.t_ACK, t, EventState.ACK);

                obj.callback = Callback(t + obj.t_DIFS, ...
                                        @()obj.NewData());
            else
                % 记录 TimeOut
                obj.all_event{end + 1} = Events(t - obj.t_ACK, t + (obj.t_TO -
obj.t_ACK), EventState.TO);

                obj.callback = Callback(t + (obj.t_TO - obj.t_ACK - obj.t_SIFS) +
obj.t_DIFS, ...
                                        @()obj.TimeOutData());
            end
        end
        % 8. 收到 ACK 事件彻底结束后
        function NewData(obj)
            % 经过的时间差
            t = obj.simulator.t;
            delta_t = t - obj.last_t;
            obj.last_t = t;
            % 让互听的 AP 进行 unhold;
            APs = obj.simulator.APs;
            for i = 1:length(obj.effect_AP)
                ap = APs{obj.effect_AP{i}};
                ap.Unhold();
            end
            assert(obj.state == ApState.SEND);
            obj.ords = 0;
            obj.t_window = obj.GetWindowTime();
```

```
            obj.state = ApState.BACK_OFF;
            obj.Run();
        end
        % 10. 超时重发 data 事件彻底结束后
        function TimeOutData(obj)
            % 经过的时间差
            t = obj.simulator.t;
            delta_t = t - obj.last_t;
            obj.last_t = t;
            % 让互听的 AP 进行 unhold;
            APs = obj.simulator.APs;
            for i = 1:length(obj.effect_AP)
                ap = APs{obj.effect_AP{i}};
                ap.Unhold();
            end
            assert(obj.state == ApState.SEND);
            obj.ords = obj.ords + 1;
            % 如果达到重传上限
            if obj.ords > obj.r
                obj.ords = 0;
                obj.t_window = obj.GetWindowTime();
                obj.state = ApState.BACK_OFF;
                obj.Run();
            else
                obj.t_window = obj.GetWindowTime();
                obj.state = ApState.BACK_OFF;
                obj.Run();
            end
        end
    end
end
```


附录 B 问题一系统理论吞吐量和冲突概率求解

B.1. 程序运行说明

通过 MATLAB 执行下面的程序，可以得到问题一场景下的系统碰撞概率和吞吐量的理论值。其中 pp 表示系统理论碰撞概率，S 表示系统理论吞吐量，S_{rts}表示启用 RTS/CTS 时系统理论吞吐量，P_{sthe}表示节点成功发送数据包概率的临界值。当$P_{\mathrm{s}} > P_{\mathrm{sthe}}$时，使用 RTS/CTS 会降低系统吞吐。当$P_{\mathrm{s}} < P_{\mathrm{sthe}}$时，使用 RTS/CTS 会提升系统吞吐

B.2. 脚本代码

```
%%
clear;
```

```
clc;
r = 32; %最大重传次数
CW_min = 16;   %最小窗口
CW_max = 1024; %最大窗口
m = log2(CW_max / CW_min);  %最多增长阶数
w0 = 16;    %初始窗口
n = 2;  %节点数量
nc = 1;     %暴露站数量
nh = 1;     %隐藏站数量
T_ack = 32;     %ACK 时间
T_sifs = 16;    %SIFS 时间
T_difs = 43;    %DIFS 时间
T_slot = 9;     %单个时隙时长
rate = 455.8;   %物理层速率(Mbps)
T_TO = 65;     % ACK TimeOut (us)
T_PHY = 13.6;   % PHY 头时长 (us)
L_MAC = 30;     % MAC 头 (Byte)
L_PLD = 1500;   % Payload (Byte)
L_RTS = 20;     % RTS 帧长
L_CTS = 14;     % CTS 帧长
%%
syms p
b00 = (2*(1-p)*(1-2*p))/(w0*(1-(2*p)^(m+1))*(1-p) + (1-2*p)*(1-
p^(r+1)+w0*2^m*p^(m+1)*(1-p^(r-m)*(1-2*p))));
tau = b00*(1-p^(r+1))/(1-p);
equation = p == 1 - (1 - tau)^(n - 1);
p_solution = vpasolve(equation, p);
for i = 1 : length(p_solution)
    if(isreal(p_solution(i)))
        pp = double(p_solution(i));
    end
end
b00 = (2*(1-pp)*(1-2*pp))/(w0*(1-(2*pp)^(m+1))*(1-pp) + (1-2*pp)*(1-
pp^(r+1)+w0*2^m*pp^(m+1)*(1-pp^(r-m)*(1-2*pp))));
tau = b00*(1-pp^(r+1))/(1-pp);
%%
Ptr = 1-(1-tau)^n;
Es = 1/Ptr - 1;
Ps = (n*tau*(1-tau)^(n-1))/Ptr;
Pc = 1 - Ps;
Ts = T_PHY + (L_MAC + L_PLD) * 8/rate + T_sifs + T_ack + T_difs;
Tc = T_PHY + (L_MAC + L_PLD) * 8/rate + T_TO + T_difs;
Tsrts = Ts + 2 * T_sifs + (L_RTS + L_CTS) * 8/rate + 2 * T_PHY;
Tcrts = L_RTS * 8 / rate + T_TO + T_difs + T_PHY;
```

```
%%% 第一问求解
S = (Ps * Ptr * L_PLD * 8)/(T_slot *(1 - Ptr) + Ps*Ptr*Ts + (1-Ps)*Ptr*Tc);
S_trs = (Ps * Ptr * L_PLD * 8)/(T_slot *(1 - Ptr) + Ps*Ptr*Tsrts + (1-
Ps)*Ptr*Tcrts);
Psthe = (Tc - Tcrts)/(Tsrts - Ts + Tc - Tcrts);
```

附录 C 问题三、四仿真图像

C.1. 问题三四理论值结果仿真说明

将仿真结果导出成.mat 文件，在以下程序中用 load()函数导出数据，系统吞吐量、RTS/CTS 机制下系统吞吐量和碰撞概率均为 3×3 的矩阵格式，用 bar()函数画出柱状图，set()函数修改横坐标名称字体和柱子颜色，最后规定 x,y 轴标签，设置不同柱子颜色对应的窗口大小和最大重传次数。

C.2. 脚本代码

```
clear;
clc;
figure('position',[150,100,900,550])%确定图片的位置和大小，[x y width height]
%准备数据
m1=load('matlab2.mat');
X=1:3;
 %画出 3 组柱状图，宽度 1
s=bar(X,m1.S,1);
 %修改横坐标名称、字体
set(gca,'XTickLabel',{'455.8Mbps','286.8Mbps','158.4Mbps'},'FontSize',10,'FontNam
e','Times New Roman');
% 设置柱子颜色
set(s(1),'FaceColor','#FA4224')
set(s(2),'FaceColor','#048243')
set(s(3),'FaceColor','#DBB40C')
%修改 x,y 轴标签
ylabel('\fontname{宋体}\fontsize{14}吞吐量(Mbps)');
xlabel('\fontname{宋体}\fontsize{14}不同物理层速率');
legend({'[W_{min}, W_{max}, r]=[16, 1024, 6]','[W_{min}, W_{max}, r]=[32, 1024,
5]','[W_{min}, W_{max}, r]=[16, 1024, 32]'})
figure('position',[150,100,900,550])%确定图片的位置和大小，[x y width height]
%画出 3 组柱状图，宽度 1
srts=bar(X,m1.Srts,1);
%修改横坐标名称、字体
set(gca,'XTickLabel',{'455.8Mbps','286.8Mbps','158.4Mbps'},'FontSize',10,'FontNam
e','Times New Roman');
% 设置柱子颜色
set(srts(1),'FaceColor','#FA4224')
```

```
set(srts(2),'FaceColor','#048243')
set(srts(3),'FaceColor','#DBB40C')
%修改 x,y 轴标签
ylabel('\fontname{宋体}\fontsize{14}RTS 机制吞吐量(Mbps)');
xlabel('\fontname{宋体}\fontsize{14}不同物理层速率');
legend({'[W_{min}, W_{max}, r]=[16, 1024, 6]','[W_{min}, W_{max}, r]=[32, 1024,
5]','[W_{min}, W_{max}, r]=[16, 1024, 32]'})
figure('position',[150,100,900,550])%确定图片的位置和大小，[x y width height]
%画出 3 组柱状图，宽度 1
p=bar(X,m1.p,1);
%修改横坐标名称、字体
set(gca,'XTickLabel',{'455.8Mbps','286.8Mbps','158.4Mbps'},'FontSize',10,'FontNam
e','Times New Roman');
% 设置柱子颜色
set(p(1),'FaceColor','#FA4224')
set(p(2),'FaceColor','#048243')
set(p(3),'FaceColor','#DBB40C')
%设置上边界确保不遮挡
ylim([0 0.55])
%修改 x,y 轴标签
ylabel('\fontname{宋体}\fontsize{14}碰撞概率 p');
xlabel('\fontname{宋体}\fontsize{14}不同物理层速率');
%修改图例
legend({'[W_{min}, W_{max}, r]=[16, 1024, 6]','[W_{min}, W_{max}, r]=[32, 1024,
5]','[W_{min}, W_{max}, r]=[16, 1024, 32]'})
```