# 实验B.4：分槽ALOHA协议仿真实验

用于在多路访问信道上确定下一个使用者的协议属于数据链路层的一个子层，称为介质访问控制（Medium Access Control，MAC）子层，许多局域网都使用多路访问信道作为它的通信基础。最简单的信道分配方案是FDM（为每个站专门分配一段频率）和TDM（为每个站分配一个时槽），当站的数量比较大而且可变，或者流量具有突发性变化的时候，就需要用到纯ALOHA协议或分槽ALOHA协议。如果信道的状态可以被检测到，可以用到1-坚持型CSMA、非坚持CSMA和P-坚持CSMA等协议。本实验模拟常见的几种MAC层多路访问协议，具体实验要求及内容如下。

【实验目的】

1. 掌握VB、VC++、VS或JAVA等集成开发环境编写仿真程序的方法；
2. 理解并掌握分槽ALOHA协议原理。

【实验内容】

编写仿真程序，对一定网络环境下MAC层的多路访问协议的分槽ALOHA协议进行实现。通过仿真，学习协议采取的介质访问管理，包括介质分配和冲突解决机制，并对协议的性能与理论结果进行比较分析。

【编程语言和环境】

1. 编程语言C/C++/C#/Java/Matlab等均可；
2. 编程环境Windows（MS Visual系列，VC/VB/VS.Net；）和Linux（编辑器vi+编译器GCC）均可；

【实验报告】

所交实验报告内容包括：

1. 实验目的与要求；
2. 实验内容与实现原理；
3. 实验具体设计实现及结果（含流程图及关键代码说明）；
4. 实验设备与实验环境；
5. 实验总结

【主要功能实现说明】

1. 仿真思路

设置各站点初始产生包的时间点及产生包的时间间隔（均为随机值），得到所有站点成功发送10000个数据包的总时间以及这段时间内所有数据包的个数（包括各站点每次新产生的包以及由于冲突而重发的包），从而计算出每包时内尝试次数及其对应的吞吐量。针对不同的包产生间隔，得到不同的每包时内尝试次数及其对应的吞吐量，将其画成一条曲线。

1. 具体步骤（示例代码为matlab）
2. 初始化各站点产生包的时间点（可采用0到1的随机数），统一归并到时槽开始的时间点。

Mgtime =[ Ttime / log(1-X/Mnum) ]\* log(rand(1,Mnum));

% 初始化各站点包产生的时间点，为[0,1]的随机数

mtime = (fix(mgtime/slot)+1) \* slot;

% 各站点数据包发送时间点，归并到时槽的开始处

其中：Ttime为发送一个包所需的时间，Mnum为站点的总个数，可通过 改变不同的X值得到不同的包产生时间点。X的取值小于站点总个数

1. 选出最早产生数据包的站点作为初始发送站点，若此时槽只有一个数据包，则发送成功；若有两个以上数据包，则冲突。记录此时槽内所有包的个数

idx = find(mtime==now\_time);

% finding of the terminal which transmission start

if length(idx) > 0

State(idx) = TRANSMIT; %State为各站点在此时槽的状态

mtime(idx) = now\_time + Mplen(idx) / Srate;

% 发送结束时间，Mplen为数据包的长度，Srate为发送的速率

mtime(idx) = round(mtime(idx)/slot) \* slot;

Tplen = Tplen + sum(Mplen(idx));

%此时槽内所有包的总长度

end

idx = find(State==TRANSMIT | State==COLLISION);

if length(idx) > 1

State(idx) = COLLISION; % 当有两个以上数据包时，发生冲突

end

1. 若成功，则发送成功的数据包数加1，程序结束点也是成功的个数为10000时。然后根据生成包的随机时间间隔，得出此站点下一次发送包的时间点。

idx = find(mtime==now\_time & State==TRANSMIT);

% finding of the terminal which transmission succeeded

if length(idx) > 0

Spnum = Spnum + 1; %发送成功的数据包数加1

Splen = Splen + Mplen(idx);%总共发送成功数据包的长度（计算吞吐量）

State(idx) = STANDBY;%设置此站点发送状态为等待

mgtime(idx) = now\_time + [ Ttime / log(1-X/Mnum) ] \* log(1-rand);

% 算出此站点下次产生的时间点

mtime(idx) = (fix(mgtime(idx)/slot)+1) \* slot;

% 归并到时槽的开始点

end

1. 若发生冲突，根据随机后退时间，得出此站点下一次发送包的时间点。

idx = find(mtime==now\_time & State==COLLISION);

% finding of the terminal which transmission failed

if length(idx) > 0

State(idx) = STANDBY;

mtime(idx) = now\_time + [ Ttime / log(1-X/Mnum) ] \* log(rand(1,length(idx)));

% 站点等待的时间，下次重发此数据包的时间点

mtime(idx) = (fix(mtime(idx)/slot)+1) \* slot; % 归并到时槽的开始处

end

1. 依次循环上述过程，直至10000个数据包成功发送。计算出每包时内传输次数及其吞吐量

traffic = Tplen / Srate / now\_time; %每包时内传输次数

%总共传输的包（包括站点新产生的包及由于冲突重传的包）的总长度除以发 送速率再除以发送完最后一个包的时间

thoughput=Splen/Srate/now\_time；%吞吐量

%成功传输的包的总长度除以发送速率再除以发送完最后一个包的时间