多路访问协议仿真

1.引言

本仿真实验对单信道多路访问协议，包括纯 ALOHA, 分槽 ALOHA, 1 持续的 CSMA, 非持续 CSMA，P 持续 CSMA 协议进行仿真，首先分析了各个协议的工作原理，然后阐述了对各个协议进行仿真的思想，最后对根据这些思想实现的仿真程序产生的结果进行分析。

2.各个协议的工作原理

纯 ALOHA 当用户有数据要发送的时候就让它们传输。当然，这样做可能会有冲突，冲突的帧将被损坏。然而，由于广播的反馈特性，发送方只要通过监听信道，总是可以知道它的帧是否被毁坏，其他的用户也可以做到这一点。在一个 LAN 中，发送方可以立即就得到反馈。如果发送出去的帧被毁坏了，则发送方只要等待一段随机的时间，然后再次发送该帧。等待的时间必须是随机的，否则的话，同样的帧会不停地冲突，因为重发的节奏完全一致。

分槽 ALOHA 与纯 ALOHA 比较，分槽 ALOHA 将时间分成离散的时间间隔，在这里将一个帧长的时间作为一个时间槽，即一个帧刚好在一个时间槽内发送完成，每个帧的发送开始时刻必须为一个时间槽开始的时刻，如果发送开始时刻不是时间槽的开始时刻，则将其延迟到下一个时间槽的开始时刻发送。那么发生冲突的时刻只能是一个时间槽的开始时刻，如果一个主机在某个时间槽的开始时刻发送数据，并且在这个时刻没有其他主机发送，则该主机一定能发送成功。如果两个以上主机在同一个时间槽的开始时刻发送数据，则等待一个大于一个帧长时间的随机时间。

持续的 CSMA 当一个站有数据要发送的时候，它首先监听信道，看当时是否有其他的站正在传输数据。如果信道忙的话，该站会一直等待直到信道空闲，当该站监测到信道空闲的时候，它就发送一帧数据。如果有冲突发生的话，该站等待一段随机的时间，然后再次监测和发送。当一个站发现信道空闲的时候，它传输数据成功的概率为 1。

非持续的 CSMA 在这个协议中，每个协议在企图传送数据之前要检测信道，如果没有人在发送数据，则该站自己开始发送数据。然而，如果信道当前正在被使用中，则该站并不持续地对信道进行监听，以便一旦前一次传输结束它好立即抓住机会发送数据。相反，它会等待一段随机的时间，然后重复同样的算法。

P-持续 CSMA 当一个站准备好要发送的数据时，它会检测信道，如果信道是空闲的，则它按照概率 p 的可能性发送数据。在概率 p=1-p 的情况下，它会将传送的数据延迟到下一个时槽。如果下一个时槽也是空闲的，则它或者传送数据，或者再次延迟，其概率分别为 p 和 q 这个过程会一直持续，直到改帧被发送出去，或者另一个站也开始传送数据。在后者的情况下，该站的处理方式如同发生了冲突一样，即等待一段随机的时候，然后再重新开始。如果该站刚开始的时候就检测到信道忙的话，它会等待到下一个时槽，然后再应用上面的算法。

3.各个协议的仿真思想

假设条件

1） 数据延迟为 0，即当同一个网络中任何一个主机发送数据时，其他主机能够马上知道。
2） 任何一个主机在它的前一个帧发送成功之前不会发送新的帧
3） 所有的主机都按相同的概率发送数据，即在同一时刻，每个主机有数据发送的概率相同
4） 所有主机发送的每个帧的长度是相同的，并且在一次仿真中是固定的。
5） 随机等待时间至少为冲突发生时刻起一个帧长的时间
6） 只有一个完整的帧的每个数据都发送成功了，才表示该帧发送成功。

通用模型

对于每一个主机，用一个数组来模拟它在各个时刻的状态，如：HostA[100]表示主机 A 在 0-99 时刻的状态，用 1 表示该时刻有数据发送，0 表示没有数据发送。如 HostA[100] = {1,1,1,1,0,0,.....}表示主机 A 在 0-3 时刻有数据发送，4-5 时刻没有数据发送，每个主机的状态都用这样的方法表示。对于数据帧长则用经过几个时刻来表示，在这个模型中，帧长的长度是固定大小的。如 HostA[100] = {1,1,1,1,0,0,.....}表示主机 A 在 0-3 时刻有数据发送，也即主机 A 在 0-3 时刻这段时间需要发送一个长度为 4 个时刻的帧。在每次仿真的开始，按照某个概率初始化每个主机在各个时刻的状态。

初始化的方法如下：

```
// 主机 X 在各个时刻以概率 p 发送数据
for(t=0; t<MAX_TIME; t++)
{
    if ( HostX[t] == x) // x 为不同于 0，和 1 的数，用来表示未经初试的状态
    {
    //若 t 时刻有数据发送，则该主机从该时刻起一个帧长的时间内都有数据发送
        概率 p    HostX[t...t+FRAME_LEN] = 1;
        概率 1-p HostX[t] = 0;
    }
}
```

假设帧长为 4 个时刻，则经初始化后 HostX 中的内容为 0 和 1 组成的字符串，其中 1 连续出现的次数为 4 的整数倍.

冲突检测：假设有 HostA,HostB，HostC 两台主机，若 HostA[t],HostB[t],HostC[t]中至少有两个值为 1，则表示在时刻 t 发生了冲突。

等 待： 某个主机需要从 t 时刻起等待 l 长的时间，进行的处理是表示该主机状态的数组中的元素，从下标为 t 的元素起整体后移 l 个元素，假设数组中原内容为“0011110111100”，需要从时刻 2 起等待 4 个时刻长度的时间，则数组中内容变为“0000001111011”,以 0 填充后移产生的空位，而超出数组长度的部分则截断。借用这个例子来说明该操作的物理意义是，该主机原预备在 2 时刻和 7 时刻各发送一个帧，由于 2 时刻发送的帧延迟到了 6 时刻发送，使得 7 时刻要发送的帧只能顺次往后延迟，但是整个时间已经不够该帧全部发送完成，按照假设条件 7),该帧发送是失败的。

纯 ALOHA 仿真

1. 引入 pos[0...UserNumber-1]记录 UserNumber 个主机正在发送的帧的起始位置
2. 引入 collision[0....UserNumber-1]记录 UserNumber 个主机在一个帧的发送过程中发生冲突的时刻，-1 表示未发生冲突，大于-1 则表示在该数表示的时刻发生了冲突。
3. 主机 n 在帧发送完毕后才会检测是否发生了冲突，若发生了冲突,则从时刻 pos[n]起等

待大于 FRAME_LEN 的随机时长。

假设主机 A,B 采用 ALOHA 协议竞争统一信道，帧长为 2, 在 10 个时刻中的数据发送情况。下面为各时刻 A,B 状态的数组中的内容以及 pos,collison 中的内容

初始状态

pos[A] = -1,pos[B] = -1, collison[A] = -1, collision[B] = -1

A	0	1	1	0	0	1	1	1	1	0
B	1	1	1	1	0	0	0	1	1	0

0 时刻

pos[A] = -1,pos[B] = 0, collison[A] = -1, collision[B] = -1

A	0	1	1	0	0	1	1	1	1	0
B	1	1	1	1	0	0	0	1	1	0

1 时刻（发生冲突）

pos[A] = 1,pos[B] = 0, collison[A] = 1, collision[B] = 1

A	0	1	1	0	0	1	1	1	1	0
B	1	1	1	1	0	0	0	1	1	0

2 时刻 （B 检测到冲突，将发生冲突的帧在 3 时刻重发）

pos[A] = 1,pos[B] = -1, collison[A] = 1, collision[B] = -1

A	0	1	1	0	0	1	1	1	1	0
B	0	0	0	1	1	1	1	0	0	0

3 时刻 （A 检测到冲突，将发生冲突的帧在 6 时刻重发）

pos[A] = -1,pos[B] = 3, collison[A] = -1, collision[B] = -1

A	0	0	0	0	0	0	1	1	0	0
B	0	0	0	1	1	1	1	0	0	0

4 时刻

pos[A] = -1,pos[B] = 3, collison[A] = -1, collision[B] = -1

A	0	0	0	0	0	0	1	1	0	0
B	0	0	0	1	1	1	1	0	0	0

5 时刻 （B 未检测到冲突，发送成功，等待下一个帧产生）

pos[A] = -1,pos[B] = 5, collison[A] = -1, collision[B] = -1

A	0	0	0	0	0	0	1	1	0	0
B	0	0	0	1	1	1	1	0	0	0

6 时刻 （发生冲突）

pos[A] = 6,pos[B] = 5, collison[A] = 6, collision[B] = 6

A	0	0	0	0	0	0	1	1	0	0
B	0	0	0	1	1	1	1	0	0	0

7 时刻（B 检测到冲突，将发生冲突的帧在 10 时刻重发）

pos[A] = 6,pos[B] = -1, collison[A] = 6, collision[B] = -1

A	0	0	0	0	0	0	1	1	0	0
B	0	0	0	1	1	0	0	0	0	0

8 时刻（A 检测到冲突，将发生冲突的帧在 12 时刻重发）

pos[A] = -1,pos[B] = -1, collison[A] = -1, collision[B] = -1

A	0	0	0	0	0	0	0	0	0	0
B	0	0	0	1	1	0	0	0	0	0

9 时刻已经没有数据要发送了，又因为 10,12 时刻已经超出了整个时间
那么，在整个 10 个时刻里只有主机 B 在 3-4 时刻内成功发送了 1 个帧。

分槽 ALOHA 仿真

与纯 ALOHA 一样，引入引入 pos[0...UserNumber-1]记录 UserNumber 个主机准备发送的帧的起始位置而不是正在发送的帧的位置，但不需要 collison 数组，因为发生冲突的时间在一个时间槽开始，是很好确定的。在每个时刻，首先检查每个主机准备发送的帧的起始位置，如果主机在该时刻准备发送数据但是该时刻不是时间槽的开始时刻，则将其延迟至下一个时间槽的开始。只有在每个时间槽的开始时刻才检测冲突。

持续的 CSMA 仿真

引入一个布尔变量 busy，当取值 true 时，表示信道忙，即已有主机在发送，反之则表示信道空闲。

引入变量 startTime 和 currentUser 分别表示当前正在使用信道的用户开始发送数据的时刻以及该用户的编号。

伪码：

```
if(!busy) //信道空闲，busy 初始化为 false
{
    if (只有编号为 n 的用户在此时刻发送 )
    {
        startTime = t;
        currentUser = n;
    }
    else if (有两个或以上的用户在此时可发送)
    {
        在该时刻发送的用户等待一段随机的时间
    }
```

```
}
else //信道忙
{
    if(t==startTime+FRAME_LEN-1) // 如果 currentUser 在这个时刻发送最后一个数据
    {
        busy = false; // 重置信道为空闲状态
    }

    if（其他用户在这个时刻要发送数据）
    {
        在这个时刻要发送数据的其他用户等待 1 个时刻 //即实现持续监听
    }

}
```

非持续的 CSMA 仿真

与持续的 CSMA 相比只有一个不同：

```
else //信道忙
{
    if(t==startTime+FRAME_LEN-1) // 如果 currentUser 在这个时刻发送最后一个数据
    {
        busy = false; // 重置信道为空闲状态
    }

    if（其他用户在这个时刻要发送数据）
    {
        在这个时刻要发送数据的其他用户等待随机个时刻 //即实现非持续监听
    }

}
```

P 持续的 CSMA 方针

与非持续的 CSMA 相比有一个不同

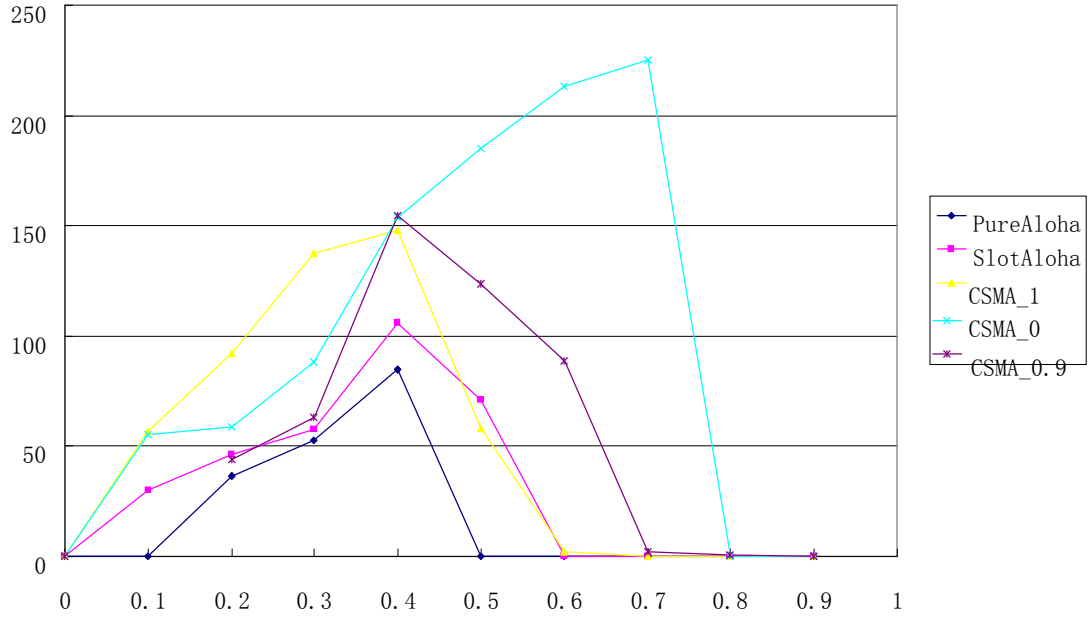
```
if(!busy) //信道空闲，busy 初始化为 false
{
    if （只有编号为 n 的用户在此时刻发送 ）
    {
        按概率 p { startTime = t;    currentUser = n;}
        按概率 1-p {用户 n 等待一个帧时再发送}
    }
    else if （有两个或以上的用户在此时可发送）
    {
        在该时刻发送的用户等待一段随机的时间
    }
```

}

4. 仿真结果及分析

按照上面所阐述的仿真算法的思想使用 c 语言在 VS2005 下编写各个协议的仿真程序。设置主机的数目为 10，帧长为 4 个时刻长，整个模拟时间为 1000 个时刻即 250 个帧长，对每个协议的仿真程序运行 10000 次。获得的数据为下图所示：

（图中横坐标表示主机发送的概率，纵坐标表示发送成功的次数）



观察上图可得出的结论及结论分析：

1. 随着发送概率的增大，采用各个协议的竞争系统在整个模拟时间内的数据发送成功的次数先增加后减少，先增加是因为随着发送概率的增加，系统有更多的数据帧要发送，提高了信道的利用率，但是当发送概率增加到一定的值时，由于系统中有过多的数据帧要发送，产生冲突的概率越大，使得发送成功的次数减少。
2. 分槽 ALOHA 与纯 ALOHA 相比，在同一发送概率下，前者成功的次数更多，并且能处理更高的发送概率，这是因为分槽 ALOHA 将发生冲突的机会减少到了纯 ALOHA 的一半左右，因此发送成功的机会更多。
3. 各种采用 CSMA 的协议的竞争系统在同一发送概率下，发送成功的次数比分槽 ALOHA 和纯 ALOHA 都要高。这是因为在采用 CSMA 协议的竞争系统中，每个主机在发送数据之前都会侦听信道，不会因为冲突使得发送的帧被破坏以及为了检测帧是否被破坏带来的延迟，一个帧一旦发送上信道则一定成功，使得信道的能够得到充分的利用，因此成功的次数更多。
4. 非持续的 CSMA 较持续的 CSMA 在发送概率较高的情况下有更高的成功概率，这是因为当发送概率较高时，持续 CSMA 将导致更大概率的冲突发生。
5. 由于 p 坚持 CSMA 在网络中主机数量多，每个主机发送概率比较高的时候工作得更好，

由于实验条件的限制，未能进行 p 取较小值的仿真，只是做了 0.9 坚持 CSMA 的仿真，实际效果观察上图可知，成功的概率比较低。但是还是比 ALOHA 协议要好。

5 结论

本仿真实验基本是成功的，得到的结果基本符合理论推导的结论。但是由于算法的设计，以及计算机工作的特性，不可能完全模拟实际的网络环境，因此产生的结果有比较明显的误差。在仿真程序的实现过程中，应为涉及频繁的数组操作，程序运行的效率也比较低，限制了仿真的规模。

网络协议仿真

实验小组组员：

084611340　李　勇

084611341　罗　敏

084611342　范学栋

084611345　唐圣潘

2009 年 4 月 16 日