Prática MAC - Redes de Computadores

Larissa Fernanda dos Santos Andrade

I. O QUE FOI PEDIDO:

Prática de simulação MAC: o aluno deve entender o Aloha puro que já está feito e desenvolver alguma coisa adicional. Como exemplos: Introduzir apenas o Carrier Sense, Introduzir o Carrier Sense 1 persistente ou que encontra um atraso aleatório para checar novamente o canal, etc. Como métricas de desempenho, o aluno pode analisar o throughput, a quantidade de bloqueio, o atraso na transmissão desde a primeira tentativa, etc.

II. ALOHA

Aloha é um medium access control (MAC) protocol para a transmissão de dados por um canal de rede compartilhado. Quando uma estação envia um pacote para outro computador, quem envia faz um broadcast com o pacote. Ao mesmo tempo, se houver qualquer outra transmissão de outros computadores, haverá uma colisão e os pacotes serão perdidos.
No Pure Aloha, sempre que uma estação tem um pacote disponível, ela o envia. Se houver colisão e o quadro for destruído, o remetente aguardará um período de tempo aleatório antes de retransmitir o pacote novamente.

III. CSMA

Quando o dispositivo deseja transmitir um pacote em um canal comum, o dispositivo começa a sentir/ouvir o canal. Se o canal estiver livre, os dispositivos transmitem o pacote; caso contrário, continuam sentindo o canal. A probabilidade de enviar o pacote é um quando o canal está livre. É por isso que é denominado como 1-Persistent CSMA.

IV. A PRÁTICA MAC

Para esta prática foi escolhido desenvolver o Carrier Sense 1 persistente e usar o throughput como métrica de desempenho.
Os arquivos disponibilizados geraram o resultado mostrado na Figura 1.
Para implementar o CSMA, a parte mudada no código dado pelo professor foi a seguinte:

```
if(ChOccEndTime <= schedule.getCurrentTime()){
    if(lastSuccPkt != NULL){
        thrp += lastSuccPkt->getDuration();
        delete lastSuccPkt;
    }
    lastSuccPkt = pkt;
    ChOccEndTime = pkt->getEndTime();
}
else{
    numPktBloq++;
    if(lastSuccPkt != NULL){
        delete lastSuccPkt;
```

```
larissa@lari-pc:~/Downloads/PráticaMAC/Pratica MAC$ g++ *.cpp -o main.exe
larissa@lari-pc:~/Downloads/PráticaMAC/Pratica MAC$ ./main.exe
Start of Simulation
Load= 0.1       Throughput= 0.0750544   PB= 0.1698
Load= 0.2       Throughput= 0.114775    PB= 0.3108
Load= 0.3       Throughput= 0.134874    PB= 0.4147
Load= 0.4       Throughput= 0.140256    PB= 0.5166
Load= 0.5       Throughput= 0.141114    PB= 0.588
Load= 0.6       Throughput= 0.12916     PB= 0.659
Load= 0.7       Throughput= 0.11921     PB= 0.7122
Load= 0.8       Throughput= 0.108907    PB= 0.7462
Load= 0.9       Throughput= 0.0981573   PB= 0.7869
Load= 1  Throughput= 0.0906263   PB= 0.8159
Load= 1.1       Throughput= 0.0835422   PB= 0.8427
Load= 1.2       Throughput= 0.075412    PB= 0.8624
Load= 1.3       Throughput= 0.0765333   PB= 0.8665
Load= 1.4       Throughput= 0.0576015   PB= 0.8992
Load= 1.5       Throughput= 0.0555659   PB= 0.9059
Load= 1.6       Throughput= 0.0480155   PB= 0.9223
Load= 1.7       Throughput= 0.044915    PB= 0.9315
Load= 1.8       Throughput= 0.0395186   PB= 0.9406
Load= 1.9       Throughput= 0.0309461   PB= 0.9513
Load= 2  Throughput= 0.0308018   PB= 0.9527
Load= 2.1       Throughput= 0.0269086   PB= 0.9599
Load= 2.2       Throughput= 0.023759    PB= 0.967
Load= 2.3       Throughput= 0.0203542   PB= 0.9721
Load= 2.4       Throughput= 0.0175736   PB= 0.9756
Load= 2.5       Throughput= 0.0166332   PB= 0.976
Load= 2.6       Throughput= 0.0154747   PB= 0.9794
Load= 2.7       Throughput= 0.011648    PB= 0.9826
Load= 2.8       Throughput= 0.0143142   PB= 0.9818
Load= 2.9       Throughput= 0.0114361   PB= 0.9848
Load= 3  Throughput= 0.00911061  PB= 0.9877
End of Simulation:
```

Fig. 1. Pure Aloha Output.

```
        lastSuccPkt = NULL;
        numPktBloq++;
    }
    ChOccEndTime = Math::MAX(ChOccEndTime, pkt
        ->getEndTime());
    delete pkt;
}
```

Como o CSMA apenas envia o pacote quando o canal está livre, não há mais pacotes bloqueados e o *else* é desnecessário. Segue abaixo minhas modificações:

```
while(ChOccEndTime > schedule.getCurrentTime()
    ){
    schedule.currentTime +=Math::Exponential(
        la);
}
if(lastSuccPkt != NULL){
    thrp += lastSuccPkt->getDuration();
    delete lastSuccPkt;
}
lastSuccPkt = pkt;
ChOccEndTime = pkt->getEndTime();
```

Além disso, para fazer uso do atributo *currentTime* da classe *schedule* foi necessário alterá-lo de *private* para *public* para que assim pudesse ser gerado um tempo aleatorio.

```
class Schedule{
public:
    Schedule();
    void initialize(TIME);
    void finalize();
    TIME getCurrentTime();
    void addEvent(Event*);
```

```
8       Event* getCurrentEvent();
9       TIME currentTime;
10  private:
11      Event *firstEvent;
12  };
```

O *while* detecta se o canal está ocupado, caso esteja, um novo tempo é gerado aleatoriamente com ajuda do metodo *Exponential* e este é comparado com o *ChOccEndTime*. Caso o canal esteja livre, o código passa do *while* e executa o *if* para fazer calculo do throughput e transmitir o pacote.
O resultado da simulação é mostrado na Figura 2.

```
larissa@lari-pc:~/Downloads/PráticaMAC/PraticaMACCSMA$ g++ *.cpp -o main.exe
larissa@lari-pc:~/Downloads/PráticaMAC/PraticaMACCSMA$ ./main.exe
Start of Simulation
Load= 0.1        Throughput= 0.090611    PB= 0
Load= 0.2        Throughput= 0.172902    PB= 0
Load= 0.3        Throughput= 0.238606    PB= 0
Load= 0.4        Throughput= 0.29972     PB= 0
Load= 0.5        Throughput= 0.362214    PB= 0
Load= 0.6        Throughput= 0.412941    PB= 0
Load= 0.7        Throughput= 0.461052    PB= 0
Load= 0.8        Throughput= 0.494441    PB= 0
Load= 0.9        Throughput= 0.546708    PB= 0
Load= 1  Throughput= 0.574056    PB= 0
Load= 1.1        Throughput= 0.608511    PB= 0
Load= 1.2        Throughput= 0.63741     PB= 0
Load= 1.3        Throughput= 0.676689    PB= 0
Load= 1.4        Throughput= 0.707493    PB= 0
Load= 1.5        Throughput= 0.729219    PB= 0
Load= 1.6        Throughput= 0.746517    PB= 0
Load= 1.7        Throughput= 0.777186    PB= 0
Load= 1.8        Throughput= 0.796841    PB= 0
Load= 1.9        Throughput= 0.820168    PB= 0
Load= 2  Throughput= 0.841915    PB= 0
Load= 2.1        Throughput= 0.850498    PB= 0
Load= 2.2        Throughput= 0.866879    PB= 0
Load= 2.3        Throughput= 0.901175    PB= 0
Load= 2.4        Throughput= 0.912287    PB= 0
Load= 2.5        Throughput= 0.925477    PB= 0
Load= 2.6        Throughput= 0.941155    PB= 0
Load= 2.7        Throughput= 0.954898    PB= 0
Load= 2.8        Throughput= 0.971432    PB= 0
Load= 2.9        Throughput= 0.974997    PB= 0
Load= 3  Throughput= 0.986529    PB= 0
End of Simulation:
```

Fig. 2. CSMA Output.

Como se pode observar o *Throughput*, que é a taxa de transferência, está bem maior com a implemnetação do CSMA. Isto ocorre porque, como já foi dito, o CSMA escuta o canal para ver se ele está livre antes de mandar um pacote, e assim, diminui as colisões.