EACH | campus capital USP LESTE

Escola de Artes, Ciências e Humanidades
Universidade de São Paulo

Nomes:	Vitor Santos Quintino	nº USP:10258880
	Thiago de Oliveira Deodato	nº USP:10258938

EP 2 - Redes de computadores

São Paulo
2019

1. Motivação da simulação

Durante as aulas, foi apresentado alguns algoritmos de acesso múltiplo. Esses algoritmos se dividem em 3 grandes grupos: os de particionamento de canal, os de acesso aleatório e os de revezamento de canal. Além desses, uma quarta opção apareceu: um algoritmo descrito como ideal, pelo slide do professor. O seu algoritmo se segue da seguinte forma:

- quando um nó quer transmitir, ele pode enviar na velocidade R.
- quando M nós querem transmitir, cada um pode enviar na velocidade média de transmissão R/M

Como nenhuma análise foi feita, surgiu a curiosidade de como alguns dos algoritmos mais usados no mercado e que estão dentro de uma das 3 categorias anteriores se saem contra esse algoritmo ideal na questão do tempo. Se são melhores ou piores veremos adiante.

2. Descrição da simulação

Todos os algoritmos foram testados de forma igual, com o mesmo processador e máquina e se utilizando da mesma linguagem, o Java.

Os algoritmos testados e simulados foram:

- Ideal;
- TDMA e FDMA de particionamento de canal;
- ALOHA puro de acesso aleatório;
- Polling de revezamento de canal.

2.1 Implementação

A cada iteração da simulação (passada como argumento da VM do Java), temos 3 passos principais:

- Gerar dados ALEATÓRIOS de pacotes;
- Simular;
- Após a simulação, calcular quanto tempo ela levou, quantos pacotes teve e de qual tamanho e adicionar em classes estatísticas.

A geração dos dados iniciais, como dito anteriormente, é aleatória. Isso significa que, antes de simular, geramos uma massa de dados com um número de pacotes aleatório, e cada pacote tem um tamanho aleatório, medido nesse programa em bytes. O número de pacotes pode variar entre 1 e 100 pacotes. O tamanho do pacote varia entre 10 a 100 MB.

Em cada algoritmo, foi escolhido um tamanho de banda padrão para realizar a transmissão dos dados. O tamanho escolhido foi o de 30.000.000 bytes (30 MBps). Este tamanho foi utilizado na primeira bateria de testes, mas ao concluirmos que os resultados poderiam ser muito diferentes com uma banda menor (como muitos temos em casa), optamos por fazer uma segunda bateria de testes com bandas de 10 MB.

2.1.1 Implementação do TDMA

Para o TDMA, alguns valores padrões também foram utilizados e que se repetem durante todos os testes. Como tamanho de janela de execução para cada pacote foi utilizado o valor de 1 segundo.

O algoritmo foi simulado calculando-se quantos “ciclos” completos de pacotes (um ciclo representaria ir da janela de tempo do ciclo 1 até a janela de tempo do último ciclo n) sendo enviados teríamos, chegando no último e calculando o tempo restante.

2.1.2 Implementação do FDMA

Para o FDMA, utilizamos 10 como o número padrão de divisões de frequência do canal.

O algoritmo foi simulado colocando os primeiros 10 pacotes dentro dos subcanais, pegando o tamanho do menor pacote e subtraindo de todos os outros.

Logo após, tiramos esse menor pacote do subcanal e colocamos um novo pacote nesse subcanal vazio, repetindo o processo. Enquanto essas subtrações estão sendo feitas, o tempo é calculado.

2.1.3 Implementação do ALOHA

Para o ALOHA, o valor padrão utilizado, o tempo para que o pacote tente ser mandado novamente, foi o de 1 segundo.

O algoritmo foi simulado tendo como premissa que a cada 1 segundo o canal tentaria mandar um novo pacote, mas daria um conflito sempre que já existisse um pacote lá. Portanto, os pacotes foram enviados e, se acabassem em um tempo quebrado, apenas no início do próximo segundo fechado o próximo pacote poderia ser enviado. Então, o tempo foi calculado seguindo esta regra e desprezando alguns milissegundos subsequentes ao fim de um pacote.

2.1.4 Implementação do Polling

Para o Polling, não tivemos nenhum valor padrão utilizado, já que é um simples algoritmos de revezamento em que um pacote vai após o outro.

O algoritmo foi simulado somando 10 a 10 o tamanho dos pacotes e calculando o tempo que eles levariam para passar pelo canal, já que neste algoritmo temos uma espécie de fila.

2.1.5 Implementação do algoritmo ideal

Para o algoritmo ideal, também não tivemos nenhum valor padrão. Foi a menor implementação de todas, embora não seja o mais simples de entender.

O algoritmo foi implementado dividindo o canal em n vezes, com n sendo o número de pacotes total da massa de dados.

3 Testes

Cada teste consistiu na execução de cada simulação um número x de vezes, calculando a média dos tempos de cada algoritmo e também calculando a média de quantidade de pacotes e a média de tamanho dos pacotes.

Para elaborar os resultados, fizemos 3 análises separadas para ter mais números de comparação: um teste com 500 simulações, um teste com 1000 simulações, e outro teste com 1500 simulações.

4 Resultados

Ao final de cada teste, é printado no console os resultados. A tabela a seguir mostra todos os resultados consolidados:

Largura de banda: 30 MB

	500 simulações	1000 simulações	1500 simulações
Média qtd. pacotes	49,226	50,489	51,792
Média tamanho pacotes	55349580 (55,34 MB)	55228044 (55,22 MB)	55126184 (55,12 MB)
Média tempo TDMA	188,8554s	193,91478s	198,76556s
Média tempo FDMA	105,14224s	107,19969s	109,31136s
Média tempo ALOHA	114,90905s	117,71202s	120,66199s
Média tempo Polling	90,82114s	92,94683s	95,170044s
Média tempo ideal	161,36461s	165,36299s	169,81071s

Largura de banda: 10 MB

	500 simulações	1000 simulações	1500 simulações
Média qtd. pacotes	50,338	50,449	49,724
Média tamanho pacotes	55063552 (55,06 MB)	55017628 (55,01 MB)	55051352 (55,05 MB)
Média tempo TDMA	494,76917s	495,9648s	488,6553s
Média tempo FDMA	319,9233s	319,56848s	316,10263s
Média tempo ALOHA	301,85928s	302,3147s	298,17996s
Média tempo Polling	277,17947s	277,5578s	273,73886s
Média tempo ideal	495,28592s	496,1341s	488,3888s

5 Conclusão

Após a análise dos resultados, foi possível notar que na verdade o algoritmo ideal não se mostrou tão performático assim em relação aos outros. Como ele divide muito o canal, grandes quantidades de dados fazem com que os subcanais fiquem muito pequenos e passe pouquíssimos dados. É um algoritmo melhor utilizado para quantidades de dados não tão grandes, e nessas condições ele possivelmente pode se mostrar melhor. Também pode ser melhor para um enlace onde dados que variem muito de quantidade passem, por ser implementado de forma dinâmica.

Entre os algoritmos de particionamento do canal, o FDMA se mostrou superior para grandes quantidades de dados. Porém, para pequenas quantidades, o TDMA pode ser superior, já que boa parte da banda do FDMA ficará ociosa enquanto o TDMA se utilizará de toda a banda. Nas condições dos testes, o FDMA levou a vantagem.

Entre o ALOHA e o Polling, o Polling se saiu melhor por pouco, e foi o melhor no geral. Por ser mais simples e só ir jogando os pacotes para serem transmitidos,

ele aproveita toda a largura de banda e transmite todos os dados de uma vez. O ALOHA faz a mesma coisa, mas tem alguns pequenos períodos de ociosidade.

Como conclusão, temos que, falando APENAS DE PERFORMANCE, o algoritmo Polling leva a vantagem. Porém, colocando outras questões no jogo, não podemos aceitar apenas o Polling como o melhor algoritmo para todos os casos.

O teste foi feito apenas com dados enormes, o que pode dar um pouco de vantagem na análise bruta do tempo. Além disso, levando em conta ambientes reais e as vantagens e desvantagens de cada algoritmo, o Polling poderia sofrer de alguns males como inanição, por exemplo. Diversos pacotes pequenos e que possivelmente podem ser importantes poderiam ter que ficar esperando por alguns poucos pacotes gigantescos e talvez não prioritários serem transmitidos. Nessa situação, outros algoritmos se sairiam melhor, como o próprio algoritmo ideal, que entregaria os pacotes pequenos muito mais rapidamente.

Portanto, tudo depende da situação que os algoritmos serão utilizados. Não existe um melhor para tudo. Nem o “ideal” é ideal em todos os casos.

6 Código

O código está hospedado no GitHub no seguinte link: https://github.com/vitorquintino/ep2_redes.

7 Referências bibliográficas

[1]
https://edisciplinas.usp.br/pluginfile.php/4636395/mod_resource/content/1/2019_Aula07_Camada_de_Enlace_de_Dados_Parte_2_2019.pdf

[2]
Slides da disciplina disponibilizados via tidia

[3]
https://www.google.com/search?q=multiple+access+methods&rlz=1C1SQJL_pt-BRBR810BR810&oq=mul&aqs=chrome.0.69i59j69i57j0l2j69i65j69i60l2j69i61.5991j0j7&sourceid=chrome&ie=UTF-8

[4]
https://www.youtube.com/