UNIVERSIDADE FEDERAL DE ALFENAS

Flaviane Moura Oliveira

2º TRABALHO PRÁTICO DE ANÁLISE DE DESEMPENHO

Alfenas/MG
2023

1. Introdução

Este trabalho consiste na elaboração de um relatório e modificação do código fonte do simulador proposto em sala de aula para coletar dados a cada 100 segundos. O objetivo será simular um roteador que recebe a chegada de pacotes de navegação web.
O tamanho dos pacotes que chegam deve atender o mais próximo possível à seguinte proporção ao término da simulação:

- 50% = 550 Bytes
- 40% = 40 bytes
- 10% = 1500 Bytes

Consideremos também os parâmetros que devem ser fixados na simulação:

A. Taxa de chegada (distribuição exponencial) com média igual a 100 pcts/s:

Temos *E[X] = 1/l, onde E[X] = 100*. Logo, ao substituirmos, fica *100 = 1/l,* resultando em *l = 1/100 = 0.01.*

Definida a taxa média de chegada, devemos calcular matematicamente o tempo de atendimento de acordo com a clássica fórmula *L/R*, onde *L* equivale ao tamanho do pacote e *R* a capacidade de atendimento do link.

Baseando-se em um link de tamanho 10000 bytes/segundo, fixado no código como TAM_LINK, temos:

A. O intervalo médio de pacotes

```
Link: 10000 bytes/segundo

10% ---> 1500
40% ---> 40
50% ---> 550

tam_medio = (0.1 * 1500) + (0.4 * 40) + (0.5 * 550) = 441
Logo, temos que o tamanho médio de pacotes é 441 bytes.
```

Figura 1: calculando intervalo médio dos pacotes

B. Capacidade do link num cenário onde a ocupação é de 60% (figura 2)

```
OCUPAÇÃO DE 60%
60% de 10000 bytes/segundo = 6000 bytes/segundo
n * 441 = 6000
sendo n = 13,60544
Consideremos que o intervalo de tempo é: 1/n
Logo a capacidade do link: 0.0735
```

Figura 2: tamanho do link (60%)

C. Capacidade do link num cenário onde a ocupação é de 80% (figura 3)

```
OCUPAÇÃO DE 80%
80% de 10000 bytes/segundo = 8000 bytes/segundo
n * 441 = 8000
sendo n = 18,14059
Consideremos que o intervalo de tempo é: 1/n
Logo a capacidade do link: 0.055125
```

Figura 3: tamanho do link (80%)

D. Capacidade do link num cenário onde a ocupação é de 95% (figura 4)

```
OCUPAÇÃO DE 95%
95% de 10000 bytes/segundo = 9500 bytes/segundo
n * 441 = 9500
sendo n = 21,54195
Consideremos que o intervalo de tempo é: 1/n
Logo a capacidade do link: 0.046421
```

Figura 4: tamanho do link (95%)

E. Capacidade do link num cenário onde a ocupação é de 99%

```
OCUPAÇÃO DE 99%
99% de 10000 bytes/segundo = 9900 bytes/segundo
n * 441 = 9900
sendo n = 22,448979
Consideremos que o intervalo de tempo é: 1/n
Logo a capacidade do link: 0.044545
```

Figura 5: tamanho do link (99%)

Calculado os valores, foi executado o simulador para os quatros cenários com a finalidade de coletar os dados finais de ocupação, medidas de little (E[N] e E[W]) e erro de little.

2. Desenvolvimento

Para atendermos a parte solicitada para coleta de dados, devemos fazer algumas modificações no código-fonte. Para isso, em primeiro lugar, devemos criar uma função responsável por coletar os dados desejados a cada x intervalo de tempo.

```
void coletaDados(little en, little ew_entrada, little ew_saida, double soma_ocupacao) {
        double en_final = (en.soma_areas) / tmp_col;
        double ew_final = ((ew_entrada.soma_areas - (ew_saida.soma_areas)) / (abs(ew_entrada.numero_eventos)));
        double lambda = ew_entrada.numero_eventos / tmp_col;



        printf("%f     | %lf", tmp_col, en_final);
        printf("        | %lf", ew_final);
        printf("        | %.15lf", en_final - lambda * ew_final);
        printf("        | %5lf\n", soma_ocupacao / tmp_col);

        tmp_col += 100;
}
```

Figura 6: função coletaDados

Após criarmos esta função, precisamos então fazer outra alteração no corpo principal do programa, que consiste em criar um novo evento responsável por coletar os dados. Podemos realizar isso ao criarmos mais um bloco '*if*' responsável pela coleta de dados sempre que o tempo decorrido desejado passe do tempo definido em ***coletaDados()***.

3. E [N]

E[N] é uma medida de desempenho na validação matemática, sabemos que E[N] é o número médio de elementos no sistema. Para calcular tal parâmetro, precisamos manipular três variáveis, sendo elas:

- *numero_eventos;*
- *soma_areas;*

- *tempo_anterior.*

```
en.soma_areas += (tempo_decorrido - en.tempo_anterior) * en.numero_eventos;
```

Figura 7: Cálculo E[N] no Simulador.c

Após definirmos isso podemos então calcular os valores de E[N] final, que são os dados que nos interessa nesse caso. Considerando os quatro cenários propostos, podemos ver no gráfico abaixo a variação de E[N] a partir da alteração no valor final da taxa de ocupação. Nota-se que quanto maior a ocupação, maior será a medida de E[N]. Para cada cenário, é possível observar que com o passar do tempo, o valor final de E [N] vai se estabilizando.

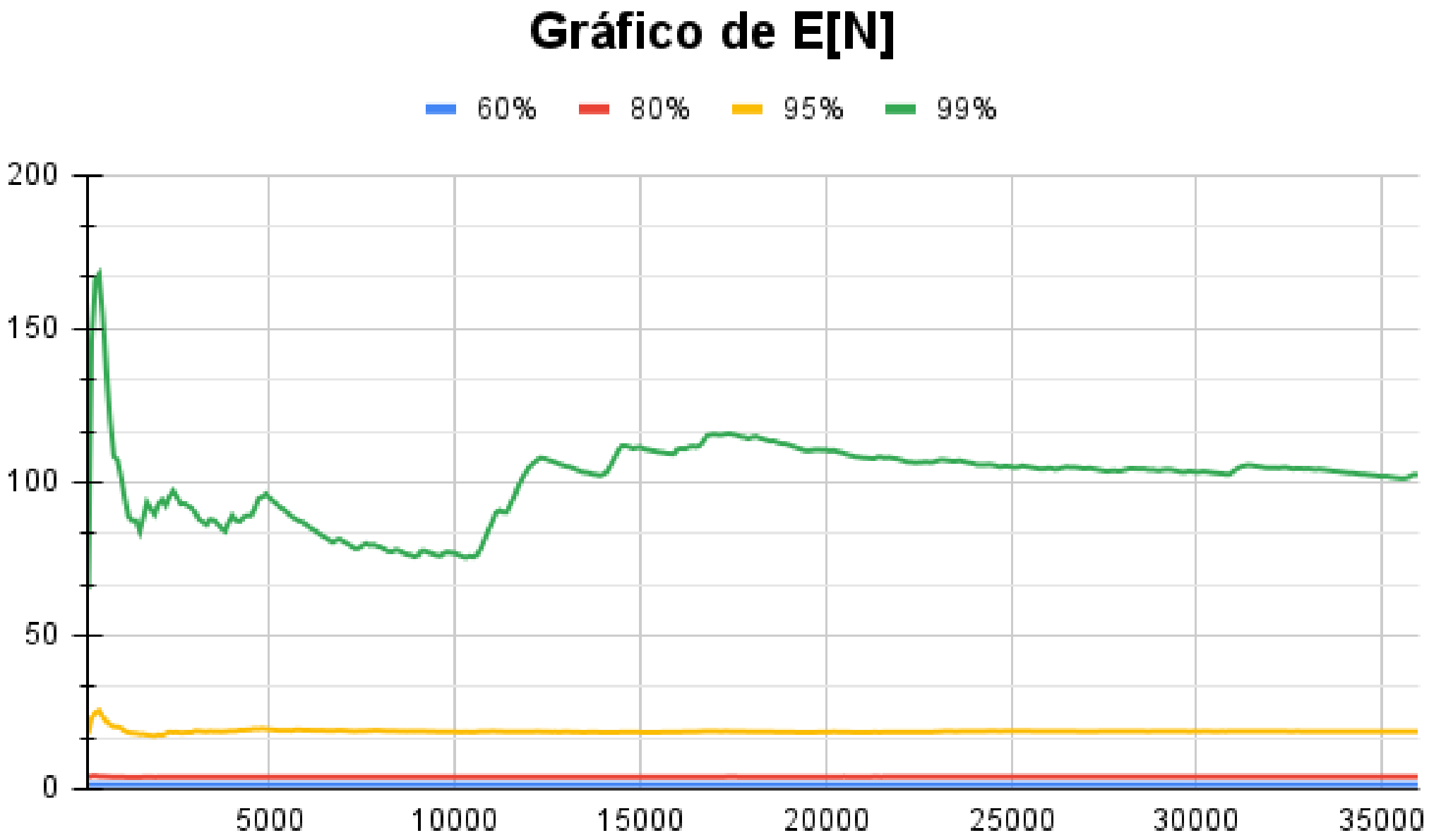
Gráfico de E[N]
60%
80%
95%
99%
200
150
100
50
0
5000
10000
15000
20000
25000
30000
35000

4. E[W]

E[W] é uma medida de desempenho na validação matemática, sabemos que E[W] é o tempo médio de espera dos elementos no sistema. Para calcular tal parâmetro, precisamos manipular duas variáveis, sendo elas:

- α (t) número de chegadas até determinado tempo (t);
- β(t) número de saídas até determinado tempo (t).

Tendo estas duas variáveis, podemos então calcular o tamanho da fila em um determinado instante (t) através da seguinte fórmula:

$$\alpha(t) - \beta(t)$$

Podemos então obter o valor de E[W].

```
ew_entrada.soma_areas +=
        (tempo_decorrido - ew_entrada.tempo_anterior) * ew_entrada.numero_eventos;
```

Figura 8: Calculando E[W] no simulador.c

Vale ressaltar que obtemos o valor de E[W] final através do cálculo da área entre a diferença do tempo médio de chegadas pelo tempo médio de saídas.

Já, para calcular o valor final de E[W] considerando os quatros cenários, assim como descrito no tópico acima, devemos pegar essa diferença e então dividir pelo tempo médio de chegadas do número de eventos.

```
double ew_final = ((ew_entrada.soma_areas - (ew_saida.soma_areas))/
                   (abs(ew_entrada.numero_eventos)));
```

Figura 9: Calculando E[W] final no simulador.c

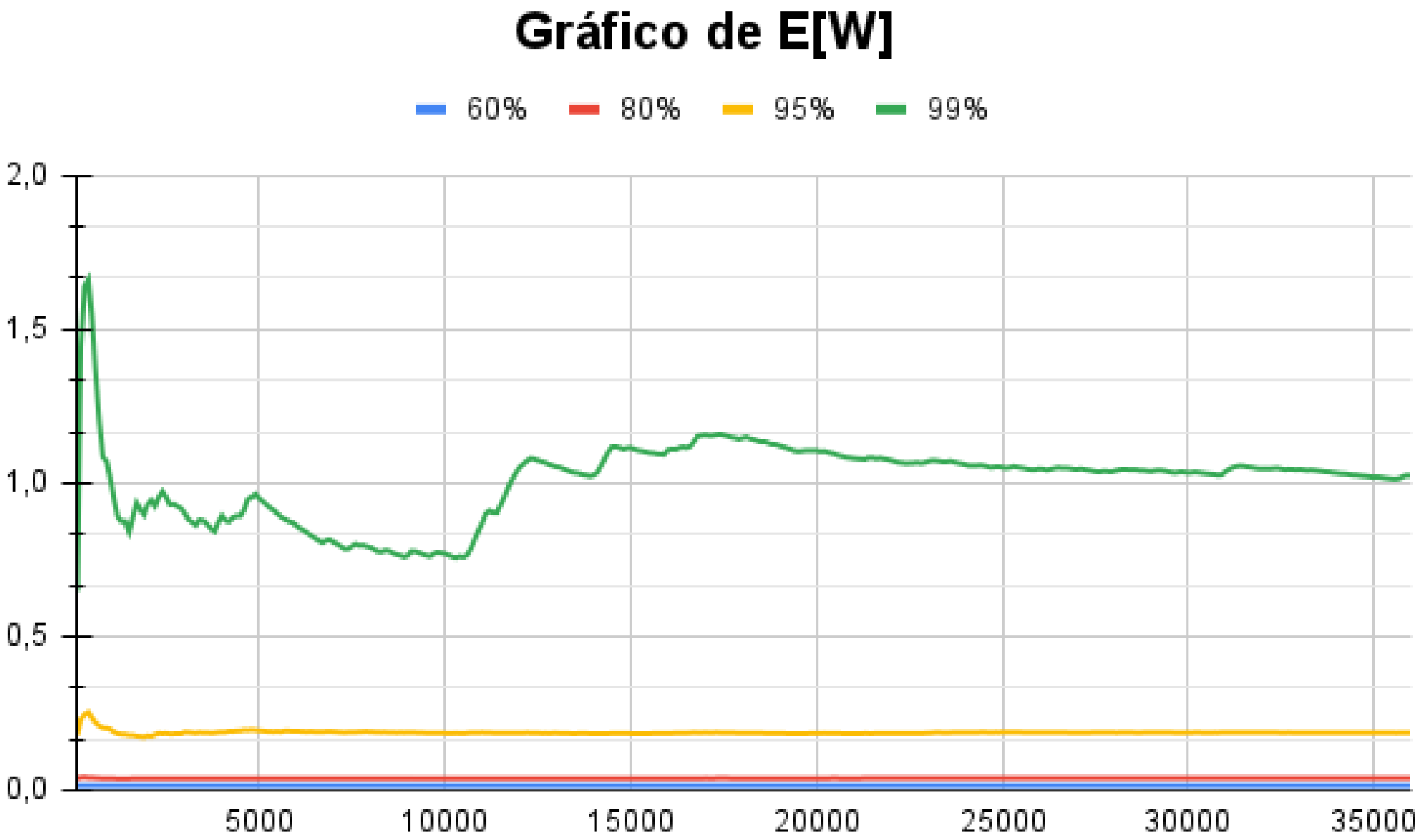
Gráfico de E[W]
60%
80%
95%
99%
2,0
1,5
1,0
0,5
0,0
5000
10000
15000
20000
25000
30000
35000

5. Ocupação

Assim como as medidas de little E[N] e E[W], a ocupação também pode ser considerada uma medida de desempenho. Mais conhecida como taxa de utilização, ela mede a fração de tempo que o sistema permanece ocupado em relação ao tempo total.

Pode-se obter matematicamente a Ocupação, calculando a chegada sobre a capacidade do sistema (λ / μ), onde:

- λ = taxa de chegada
- μ = taxa de saída

É importante ressaltar que o valor da taxa de utilização nunca pode exceder 1, visto que, se ocorrer, teremos um crescimento infinito de fila.

Calculada a taxa de utilização final para todos os cenários, podemos observar que: quanto maior a ocupação mais ela se aproxima de seu valor máximo, 1. Assim como podemos notar uma estabilização ao decorrer do tempo de serviço.

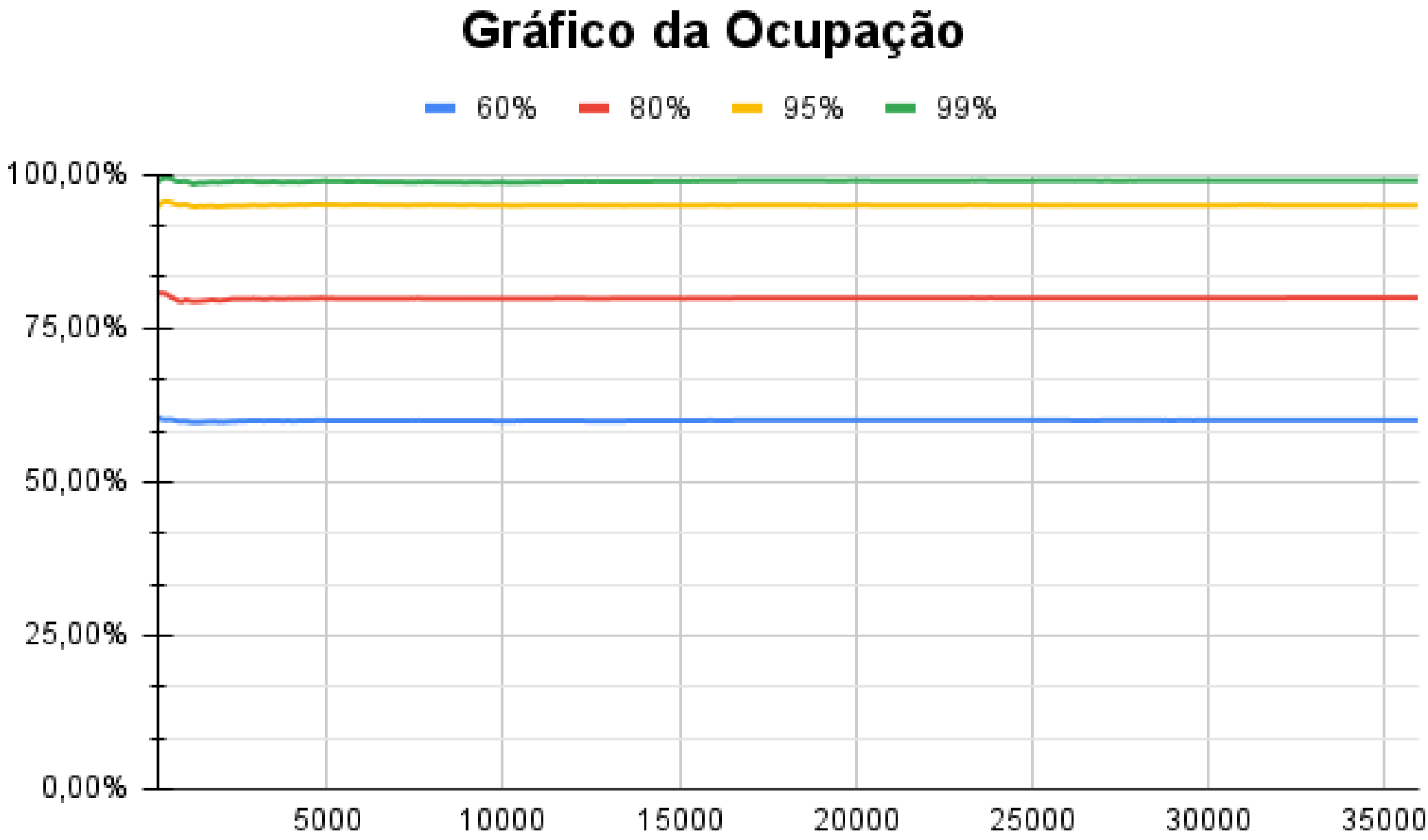
Gráfico da Ocupação
60%
80%
95%
99%
100,00%
75,00%
50,00%
25,00%
0,00%
5000
10000
15000
20000
25000
30000
35000

6. Erro de Little

Nota: para conseguir apresentar a variação do Erro de Little em um gráfico, tive que pegar os valores absolutos, convertendo-os manualmente, pois mesmo chamando a função ***abs()*** da biblioteca *math.h* para fazer essa conversão, ao rodar o algoritmo, muitos dos valores ficaram negativos, como pode-se verificar nos arquivos txt enviados com o código-fonte do simulador.


Gráfico de Erro de Little
60%
80%
95%
99%
1,25E+08
1,00E+08
7,50E+07
5,00E+07
2,50E+07
0,00E+00
5000
10000
15000
20000
25000
30000
35000