Universidade Federal do Rio de Janeiro

Simulação de Filas

Trabalho de Avaliação e Desempenho - MAB 515
Professor:Paulo Henrique de Aguiar Rodrigues

RIO DE JANEIRO
2011

Integrantes e suas Participações:

- Diego Fonseca Pereira de Souza DRE:108055513
  - Assinatura: _____
  - Participação:
    * Obtenção de Resultados
    * Confecção do Relatório
    * Confecção dos Gráficos
    * Teste de Validação
    * Otimização
    * Análise de Resultados
- Ewerton Vinicius Ramos Balthazar DRE:106044875
  - Assinatura: _____
  - Participação:
    * Obtenção de Resultados
    * Confecção do Relatório
    * Teste de Validação
    * Otimização
    * Análise de Resultados
- Gustavo Rodrigues Lima DRE:108055416
  - Assinatura: _____
  - Participação:
    * Implementação do Simulador
    * Depuração Do Simulador
    * Confecção do Relatório
    * Teste de Validação
    * Estimativa da fase transiente
    * Estimativa de tamanho e número de rodadas
- Gustavo Daniel Soares Figueiredo DRE:106051068
  - Assinatura: _____
  - Participação:
    * Implementação do Simulador
    * Depuração Do Simulador
    * Confecção do Relatório
    * Teste de Validação
    * Estimativa da fase transiente
    * Estimativa de tamanho e número de rodadas

1 Apresentação do Problema

Executar uma simulação do comportamento de um sistema no qual duas filas disputam o servidor e umas das filas tem prioridade preemptiva sobre a outra.

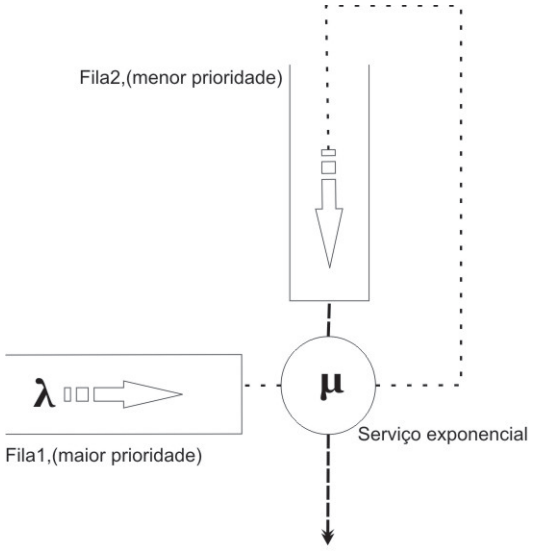

Figura 1: Esquema do sistema

Fregueses chegam à fila 1 segundo um Processo Poisson com taxa λ (tempo entre chegadas é exponencial com taxa λ).Esta é a fila de maior prioridade do sistema.Após serem servidos pela primeira os fregueses seguem para a fila 2 , de menor prioridade .Ao término deste segundo serviço, os fregueses vão embora.Tanto o primeiro como o segundo serviços do freguês são obtidos de forma independente a partir de uma distribuição exponencial com taxa $\mu = s^{-1}$.Isto significa que os serviços recebidos por um mesmo freguês são totalmente independentes.Todavia,o serviço em andamenteo da fila 2 pode ser interrompido ,pela chegada de um freguês na fila 1.Neste caso, o serviço interrompido será retomado de onde parou.Observe que um freguês da fila 2 poderá ser interrompido mais de uma vez.As filas operam sobre o regime **FCFS** (First Come First Servide - o primeiro à chegar é o primeiro à ser servido).

2 Solução Proposta

2.1 Introdução

2.1.1 Funcionamento Geral do Simulador

Quando o programa é inicializado, ele chama a classe Simulador esta chama a ManipulaXML que farará a leitura do xml, este arquivo contém os parâmetros que podem ser alterados do programa , como o tamanho da fase transiente número de rodadas , tamanaho da rodada, disciplina de atendimento , utilização e taxa de serviço.Após a leitura do xml a classe Simulador cria o "loop"principal do programa, baseado no número de rodadas , a cada rodada ela chama a classe

Rodada que cria as filas de acordo com a disciplina especificada.A partir desse momento começa-se a gerar os eventos de chegada e saída do programa , mas para fazer isso fazemos uso da Classe Gerenciado de Eventos.O objetivo da classe GerenciadorEventos é gerar os eventos e dizer qual é o próximo evento a ser tratado , caso o próximo evento seja de chegada e o seridor não esta ocupado o novo freguês vai direto para o serviço , no nosso sistema a classe Servidor que realiza o serviço, caso o servidor esteja ocupado com um freguês oriundo fa fila 2, o mesmo é retirado do servidor e reposicionado na fila com o tempo de serviço residual preenchido , o último caso possível é quando quem está em serviço é um frêgues da fila 1 , logo o novo cliente terá de ficar esperando na fila.

2.1.2 Os Eventos Escolhidos

O sistema possui três eventos:
Chegada→ Ocorre quando um cliente chega no sistema, ou seja um novo freguês entra na fila 1 ou direto para serviço
Saida→ Ocorre quando um cliente sai da sua fila e vai para o servidor,se o freguês vem da fila 1, quando acabar de ser sevido vai para a fila 2 , caso seja da fila 2 irá embora do sistema.
Interrupção→ Quando ocorre um evento de chegado no sistema e quem está em serviço é um freguês da fila 2, efetivamente não um evento de interrupção, porque depois de tratado virá um evento de saida.O que fazemos é devolver o clinte interrompido para sua fila e colocar o tempo de serviço residual em um variável.

2.1.3 Método usado:replicativo

Este método se configura da seguinte forma: a cada rodada de simulação usa-se uma nova semente.Deve ter bastante cuidado com a estimativa da fase transiente e ter sementes bastante distantes uma das outras para se evitar qualquer dependência estocástica entre as medidas.Para gerar semente nos chamamos a classe Math do java e pedimos para ela gerar um número aletório , após isso multiplicamos por um número muito grande para aumentar a probabilidade de sempre termos sementes distintas, com a semente gerada pegamos a mesma e passamos como parâmetro para classe Random do java, para agora sim podemos gerar nosso eventos.

2.1.4 Geração da variáveis aleatórias

Para gerar as variáveis aleátorias fazemos uso da seguinte fórmula $x_0 = -\frac{\ln u_0}{taxa}$, onde u_0 é gerado aleatoriamente pela classe GeradorAleatório e a taxa depende para qual evento estamos querendo gerar uma amostra. Caso seja para um evento de saida usaremos μ que representa a taxa de serviço.Já se for evento de chegada usaremos λ, que é a taxa com que os fregueses chegam ao sistema , este valor pode ser obtido pela seguinte fórmula: $\rho = 2\lambda$, onde ρ representa a utilização do sistema que é informado no arquivo de configuração.

2.1.5 Escolha dos Parâmetros

2.1.6 Implementação do Conceito de Cores

O método replicativo gera uma facilidade na implementação das cores, basta apenas dizer se o freguês faz ou não parta da fase transiente.Criamos um "enumerator", faz com um variável possa apenas possa assumir os valores informado no "enumerator", que no nosso caso será "TRANSIENTE"ou "NAOTRANSIENTE"dessa forma implementamos o conceito de cores.A diferença entre a marcação é que quando marcamos um cliente como "TRANSIENTE"ele não será levado em consideração no cálculo estatístico.A fase transiente sever basicamente para colocar o sistema em equilíbrio.

2.1.7 Estrutura Interna Utilizada

As estruturas utilizadas para representar as filas foram as seguintes , para as com disciplina de atendimenteo FCFS (First Come First Servide) fazemos uso de uma fila e sempre inserimo um novo cliente no final, quando o servidor está desocupado e tem cliente na fila pegamos o primeiro da fila.Já a disciplina LCFS(Last Come First Servide) colocamos o cliente no inicio da fila , se o servidor está vazio e tem freguês na pilha colocamos o primeiro da pilha em serviço.

2.1.8 Equimamentos de Teste

O equipamento utilizado nos teste foi o seguinte:

- Processador: i5 - Modelo 760 - Clock de 2,8 Ghz
- Memória: 2x2GB DDR3 - Modelo:KVR1333D3N9 - Trabalhando em Dual Channel
- Placa-Mãe: Gigabyte - Modelo:P55-USB3
- Sistema Operacional: Windows 7 Professional 64 bits

2.1.9 Linguagem Utilizada

A linguagem escolhida foi o Java , devido ao maior dominio dos integrantes do grupo nesta linguagem e também pelas estruturas de alta performance já implementadas.

2.1.10 Outra Informações Pertinentes

1. Para poder rodar o programa,se for o windows criamos o arquivo simulador.bat que tem internamente o comando para rodar o arquivo .jar ,gerado do código do programa, caso seja linux criamos o simulador.sh.
2. O programa pode tratabalhar tanto com a disciplina de atendimenteo FCFS ou LCFS.
3. Para não precisar ir ao código para trocar os parâmetros de entrada nos criamos um arquivo chamado config.xml que fica dentro da pasta XML, este arquivo sempre será lido quando se inicia o simulador , caso não seja

possível localizar o arquivo, a simulação será realizada com os parâmetros padrão especificado no código.

```
<ad>
    <configuracao>
        <fasetransiente>1400</fasetransiente>
        <numerorodadas>30</numerorodadas>
        <tamanhorodadas>14000</tamanhorodadas>
        <taxaservico>1.0</taxaservico>
        <utilizacao>0.8</utilizacao>
        <fila1>FCFS</fila1>
        <fila2>FCFS</fila2>
    </configuracao>
</ad>
```

Figura 2: Exemplo do config.xml

Explicação de cada tag:
fasetransiente→Determina a quantidade de chegadas(Fregueses) que não terão estatísica computada no início de cada rodada. O objetivo é que o sistema alcance o equilíbrio para ter maior exatidão mo cálculo estatístico.
numerorodadas→Determina quantas rodadas serão simuladas, computando as estatísticas ao final de cada uma delas.
taxaservico→Taxa com que o servidor ira atender os fregueses.
tamanho da rodada→Número total de chegadas que ocorrerão e serão tratadas pelo sistema em uma rodada. utilizacao→taxa que indica a utilizacao do sistema ou seja ρ.
fila1 e fila2→indica a disciplina de atendimenteo na fila 1 e 2, respectivamente.Podendo ser FCFS ou LCFS.

2.2 Teste de Correção

Para verificar a correção do simulador usaremos de dois casos básicos:

- Quando ocorre uma única chegada no sistema:
- Quando o cliente chega ao sistema e encontra fregueses na fila 1, 2 e fregueses sendo servidos;

Vamos descrever passo a passo o comportamento esperado do simulador e, em seguida, verificar o seu funcionamento no modo debug.
Primeiro caso – quando o cliente típico chega ao sistema e encontra todas as filas e serviço vazios, não ocorrendo chegadas depois dele:

- **Tempo = 5**
  Nesse instante chega o cliente típico ao sistema e não encontra ninguém. Logo ele é encaminhado para o serviço e seu tempo de espera na fila 1 é zero.
- **Tempo = 6**
  Nesse instante ocorre o término do serviço do cliente e ele é encaminhado para a fila 2. Porém, a fila está vazia da mesma forma que o serviço.

Portanto , o cliente é encaminhado para o atendimento e seu tempo de espera na fila 2 é zero.

- **Tempo = 7**
  Nesse instante ocorre o término do serviço do cliente e ele deixa o sistema. O tempo gasto na fila 1 e no primeiro serviço (T1) é de uma unidade de tempo e o tempo gasto na fila 2 e no segundo serviço (T2) é de uma unidade de tempo, totalizando duas unidades de tempo de permanência no sistema(T).

Na execução do simulador, com parâmetros $\rho = 0.8$ ($\lambda = 0.4$), tamanho da fase transiente igual zero, tamanho da rodada e número de rodadas iguais a um. Temos o seguinte:

- **Tempo = 0**
  É gerado um evento de chegada para o tempo T = 4,70914977913353.

- **Tempo= 4,70914977913353**
  O cliente típico chega na fila 1, encontrando-a vazia. Logo, ele é servido e um evento de saída do primeiro serviço é gerado para o tempo T = 5,408822159693859.

- **Tempo= 5,408822159693859**
  O cliente típico é retirado do serviço. Como a fila 2 está vazia e o serviço ocioso (lembrando que só ocorre uma chegada no sistema), ele retorna ao servidor para executar o segundo serviço. Nesse momento é gerado um evento de saída para T = 6,670187558040615.

- **Tempo= 6,670187558040615**
  O cliente típico é retirado do serviço e sai do sistema. O resultado impresso ao final da execução é mostrado abaixo:

E[W1]	0.0
E[Nq1]	0.0
E[T1]	0.6996723805603287

Tabela 1: Resultado Da Fila 1

E[W2]	0.0
E[Nq2]	0.0
E[T2]	1.261365398346756

Tabela 2: Resultado Da Fila 2

E[N]:0.294000395317691

No segundo caso – em um instante de tempo qualquer em que exista alguém na fila 1 ou 2 e alguém em serviço, vamos analisar o comportamento esperado e apresentado pelo simulador. Digamos que a fila 1 esteja vazia, exista um cliente na fila 2, um cliente da fila 2 em serviço e um cliente chega ao sistema.

- **Tempo = 105**
  A fila 1 está vazia, a fila 2 possui um cliente em espera e um outro cliente vai para o servidor, nesse momento (T = 105), executar seu segundo serviço. Então, o segundo serviço terminará em T = 106.

- **Tempo = 105,3**
  Nesse momento chega um cliente no sistema. A fila 1 está vazia e, como no serviço existe um cliente da fila 2, este é retirado do serviço, adicionado no início da fila 2 e o freguês que chegou ao sistema em T = 105,3 vai para o servidor executar o primeiro serviço.

- **Tempo = 106,3**
  Nesse momento termina a execução do serviço do cliente da fila 1. Ele é, então, adicionado ao final da fila 2 (que possui três clientes). Como a fila 1 está vazia, o primeiro cliente da fila 2 é adicionado ao servidor. Esse cliente foi interrompido por uma chegada, possuindo serviço residual igual a 0,7. Logo, só falta executar esse tempo de serviço para esse cliente.

- **Tempo = 107**
  O cliente tem seu serviço completo e sai do sistema. Como a fila 1 está vazia, o primeiro cliente da fila 2 vai para o servidor para realizar seu segundo serviço.

- **Tempo = 107,9**
  Nesse momento ocorre uma chegada no sistema. O cliente em serviço, que é da fila 2, é interrompido e adicionado no início da fila 2 com serviço residual 0,1 e o cliente que chegou no sistema vai para o servidor para executar o seu primeiro serviço.

- **Tempo = 108,9**
  O freguês da fila 1 sai do serviço e é adicionado ao final da fila 2. Como a fila 1 está vazia, o primeiro cliente da fila 2 é vai para o servidor executar o restando de seu serviço (resíduo de 0,1).

- **Tempo = 109**
  O cliente tem seu serviço completo e sai do sistema. Em seu lugar é adicionado o primeiro cliente da fila 2, visto que a fila 1 está vazia.

- **Tempo = 110**
  O cliente tem seu serviço completo e sai do sistema. No seu lugar é adicionado o último cliente da fila 2.

- **Tempo = 111**
  O último cliente tem seu serviço completo e deixa o sistema.

O que queremos frisar com esse passo a passo é que, à medida que os clientes chegam ao sistema, eles são adicionados a fila 1. Se, por ventura, a fila 1 estiver vazia e um cliente da fila 2 estiver sendo servido, este último tem seu serviço interrompido para dar lugar ao cliente que acabou de chegar no sistema. Quando a fila 1 é esvaziada (todos os clientes são servidos pela primeira vez e adicionados ao final da fila2), o cliente interrompido anteriormente retoma o seu serviço e continua-o de onde parou. Apresentamos agora o comportamento do simulador durante a execução com parâmetros $\rho = 0.8$ ($\lambda = 0.4$), tamanho

da fase transiente igual 2000, tamanho da rodada igual a 20000 e número de rodadas igual a 20.

- **Tempo = 22,529191964820306**
  Um cliente chega ao sistema e encontra as filas vazias e serviço ocioso. Ele é imediatamente levado ao servidor para executar o primeiro serviço. O término foi estimado para T = 23,316849360653332.

- **Tempo = 22,609431733190895**
  Uma nova chegada ocorre. O cliente é adicionado na fila 1, visto que o existe um cliente da fila 1 em serviço.

- **Tempo = 23,316849360653332**
  O cliente da fila 1 termina o primeiro serviço e é adicionado a fila 2. O servidor recebe o cliente da que estava em espera na fila 1. O tempo estimado para o término do serviço é T = 24,93281846492645.

- **Tempo = 24,93281846492645**
  O cliente da fila 1 termina o primeiro serviço e é adicionado ao final da fila 2. O servidor recebe o cliente que estava em espera na fila 2, visto que a fila 1 estava vazia. O término do serviço foi estimado para T = 27,0134184578530564

- **Tempo = 25,191930293792005**
  Nesse momento chega um cliente no sistema. A fila 1 está vazia, mas existe um cliente da fila 2 em serviço. O cliente que está em serviço é interrompido e adicionado ao início da fila 2 com serviço residual igual a 1,8214881640610514 e o cliente que chegou no sistema vai para o servidor executar o primeiro serviço. O tempo estimado para o término do serviço é T = 25,361633546604896.

- **Tempo = 25,361633546604896**
  O serviço do cliente da fila 1 é terminado. Ele é, então, adicionado ao final da fila 2 e o servidor recebe o cliente que havia sido interrompido por uma chegada no sistema.

- **Tempo = 25,781834055864483**
  Uma nova chegada ocorre no sistema. A fila 1 está vazia, mas existe um cliente da fila 2 em serviço. O cliente que está em serviço é adicionado ao início da fila 2 com serviço residual igual a 1,4012876548014646 e o cliente que chegou no sistema vai para o servidor executar o primeiro serviço. O tempo estimado para o término do serviço é T = 26,09284250891023.

- **Tempo = 26,09284250891023**
  O serviço do cliente da fila 1 é terminado. Ele é, então, adicionada ao final da fila 2 e o servidor recebe o cliente que havia sido interrompido por uma chegada no sistema (vale ressaltar que ele foi interrompido duas vezes).

- **Tempo = 27,494130163711695**
  O serviço do cliente da fila 2 é terminado. O primeiro cliente que está esperando na fila 2 é, então, enviado ao servidor, visto que a fila 1 está vazia. O tempo estimado para o término do serviço é T = 27,67590336255582.

- **Tempo = 27,67590336255582**
  O serviço do cliente da fila 2 é terminado. O primeiro cliente que está esperando na fila 2 é, então, enviado ao servidor, visto que a fila 2 está vazia. O tempo estimado para o término do serviço é T = 28,55821476725657.

- **Tempo = 28,55821476725657**
  O serviço do cliente da fila 2 é terminado. O único cliente que está esperando na fila 2 é, então, enviado ao servidor, visto que a fila 2 está vazia. O tempo estimado para o término do serviço é T = 28,77942028999687.

E assim o sistema continua suas iterações. Cada chegada da fila 1 (prioritária) interrompe o serviço em execução de clientes da fila 2. E os clientes interrompidos retomam seus serviços do ponto em que pararam (continuidade).

E[W1]	0.6646793255388631		
Tamanho do IC	0.5655075926736065%		
Mínimo	0.6609205134860091	Máximo	0.6684381375917171
E[T1]	1.663596027805336		
Tamanho IC	0.2755513216285042%		
Mínimo	1.6590119669641592	Máximo	1.668180088646513

Tabela 3: Resultado Da Fila 1

E[W2]	10.716336150492388		
Tamanho do IC	1.436850390620103%		
Mínimo	10.562358432653875	Máximo	10.870313868330902
E[T2]	11.716034940830149		
Tamanho IC	1.3189525919822118%		
Mínimo	11.561505994300528	Máximo	11.870563887359769

Tabela 4: Resultado Da Fila 2

Desta forma, provamos que o simulador funciona corretamente.

2.3 Estimativa da Fase Transiente

Como decidimos utilizar o método replicativo no simulador, um dos parâmetros extremamente importante a ser estimado é o tamanho da fase transiente. A fase transiente determina quantas chegadas devem ocorrer antes de coletarmos os dados de cada cliente que chega ao sistema. Ou seja, só coletaremos os dados quando o sistema estiver em equilíbrio, nesse caso a taxa de entrada é igual à taxa de saída.Para determinarmos o tamanho da fase transiente, executamos o simulador um milhão de vezes com sementes diferentes para cada valor de utilização (0.2, 0.4, 0.6, 0.8 e 0.9). E calculamos o número médio de chegadas e desvio padrão quando o simulador apresentava taxa de chegada entre 95% e 105% do valor real. Desta forma, garantimos a independência do tamanho da fase transiente e o valor da semente.Calculada a média e o desvio padrão, adotamos que o tamanho da fase transiente é o valor da média acrescido de dois desvios padrão. No caso de $\rho = 0.9$, após um milhão de iterações com sementes diferentes, encontramos 843 como média e 863 como desvio padrão. Então, o

tamanho da fase transiente é 2569, mas adotamos o valor de 2600. No caso de $\rho = 0.2$, após um milhão de iterações com sementes diferentes, encontramos 843 como média e 864 como desvio padrão. Então, o tamanho da fase transiente é 2571, mas adotamos o valor de 2600. Abaixo, ilustraremos apenas o caso em que o valor da utilização é de 0.9, onde utilizaremos o tamanho da fase transiente encontrada e um tamanho abaixo do que foi determinado.

ρ = 0.9 fase transiente = 2600

E[W1]	0.6668867754192386		
Tamanho do IC	0.308757023742038%		
Mínimo	0.6648277156597249	Máximo	0.6689458351787523
V(W1)	1.7772825476186607		
Tamanho IC	0.6518506784297107%		
Mínimo	1.7656973192743957	Máximo	1.7888677759629257
E[T1]	1.667136648944126		
Tamanho IC	0.1462170202162575%		
Mínimo	1.6646990114131066	Máximo	1.6695742864751453
E[Nq1]	0.26672946487184795		
Tamanho IC	0.3381780488824411%		
Mínimo	0.26582744437174977	Máximo	0.26763148537194614
E[N1]	0.666787634433624		
Tamanho IC	0.17641575106640983%		
Mínimo	0.66561131602032	Máximo	0.667963952846928

Tabela 5: Resultado Da Fila 1

E[W2]	10.661271201935511		
Tamanho do IC	0.7280503809410517%		
Mínimo	10.58365177633666	Máximo	10.738890627534362
V(W2)	149.47419425388307		
Tamanho IC	2.134893135433103%		
Mínimo	146.28307994151297	Máximo	152.66530856625317
E[T2]	11.660943735779357		
Tamanho IC	0.6674550306110023%		
Mínimo	11.58311218019818	Máximo	11.738775291360534
E[Nq2]	4.264172792211398		
Tamanho IC	0.7576415269944475%		
Mínimo	4.231865648354806	Máximo	4.29647993606799
E[N2]	4.664000007549481		
Tamanho IC	0.6971228544277265%		
Mínimo	4.631486197566343	Máximo	4.696513817532619

Tabela 6: Resultado Da Fila 2

Note que o intervalo de confiança está dentro do valor pedido. Agora, vamos executar a mesma simulação utilizando, apenas 100 chegadas transientes.

ρ = 0.9 fase transiente = 100

E[W1]	0.6662495898690719		
Tamanho do IC	0.7206529175717239%		
Mínimo	0.6614482427613708	Máximo	0.671050936976773
V(W1)	1.7805096725369913		
Tamanho IC	1.5261018433114621%		
Mínimo	1.7533372816040653	Máximo	1.8076820634699173
E[T1]	1.6658719820223422		
Tamanho IC	0.3528494785355549%		
Mínimo	1.6599939614207064	Máximo	1.671750002623978
E[Nq1]	0.266454793597033		
Tamanho IC	0.7736720311467413%		
Mínimo	0.264393307383323	Máximo	0.268516279810743
E[N1]	0.6662194681138336		
Tamanho IC	0.40509503497381955%		
Mínimo	0.6635206461264754	Máximo	0.6689182901011917

Tabela 7: Resultado Da Fila 1

E[W2]	10.724994669241422		
Tamanho do IC	1.4741937636863396%		
Mínimo	10.566887466671773	Máximo	10.883101871811071
V(W2)	154.0179169952991		
Tamanho IC	5.380383848651983%		
Mínimo	145.7311618652538	Máximo	162.30467212534438
E[T2]	11.725104866123845		
Tamanho IC	1.3507224309141284%		
Mínimo	11.566731244648906	Máximo	11.883478487598783
E[Nq2]	4.289552180828222		
Tamanho IC	1.534270167253853%		
Mínimo	4.2237388614089886	Máximo	4.355365500247457
E[N2]	4.68951306935598		
Tamanho IC	1.4111564411308977%		
Mínimo	4.6233367036200885	Máximo	4.755689435091872

Tabela 8: Resultado Da Fila 2

Nesse último caso, o intervalo de confiança da variância de W2 ficou um pouco acima do desejado.

2.4 Tabelas com os resultados

Executamos o simulador usando como parâmetros os seguintes valores:

- tamanho da fase transiente = 2600
- tamanho da rodada = 26000
- número de rodadas = 50

É possível obter resultados razoáveis para as medidas de eficiência com valores menores do que os utilizados acima para algumas utilizações. Resolvemos, portanto, utilizar tais valores para a simulação do sistema com as taxas de utilização 0.2, 0.4 e 0.6. Quando a utilização é 0.8, utilizamos o tamanho da rodada como 100000 e, para utilização de 0.9, utilizamos o tamanho da rodada como 200000. A quantidade de eventos gerados foi maior devido à variância de W2 que apresentava intervalo de coeficiente fora do tamanho pedido. Devido a este fato, os valores E[W2], E[T2], E[Nq1] e E[N2] apresentavam erros significativos. Quando aumentamos a quantidade de eventos de chegadas a serem computados, conseguimos não apenas diminuir o intervalo de confiança de V(W2), mas aproximamos os valores encontrados no simulador com os valores calculados de maneira empírica. Também diminuímos o intervalo de confiança de todas as medidas de desempenho.

ρ	0.2	0.4	0.6	0.8	0.9
$\rho 1$	0.1	0.2	0.3	0.4	0.45
$\rho 2$	0.1	0.2	0.3	0.4	0.45
μ	1	1	1	1	1
λ	0.1	0.2	0.3	0.4	0.45
E[X]	1	1	1	1	1
$E[X^2]$	2	2	2	2	2
E[Xr]	1	1	1	1	1
E[W1]	0.11111111111111	0.25	0.42857142857143	0.66666666666667	0.81818181818182
E[T1]	1.11111111111111	1.25	1.42857142857143	1.66666666666667	1.81818181818182
E[W2]	0.52777777777778	1.5	3.64285714285714	10.6666666666667	25.3636363636364
E[T2]	1.52777777777778	2.5	4.64285714285714	11.6666666666667	26.3636363636364
E[Nq1]	0.011111111111111	0.05	0.12857142857143	0.26666666666667	0.36818181818182
E[Nq2]	0.052777777777778	0.3	1.09285714285714	4.26666666666667	11.4136363636364
E[N]	0.26388888888889	0.75	1.82142857142857	5.33333333333334	12.6818181818182
E[N1]	0.11111111111111	0.25	0.42857142857143	0.66666666666667	0.81818181818182
E[N2]	0.15277777777778	0.5	1.39285714285714	4.66666666666667	11.8636363636364

Tabela 9: Valores Esperados

ρ	0.2	0.4	0.6	0.8	0.9
E[W1] esperado	0.111111111	0.25	0.428571428	0.666666667	0.8181818182
E[W1] simulado	0.11089977474521	0.25018511550992	0.42746276847349	0.66797431532864	0.81717536230335
IC máximo	0.11191831196347	0.25209499587169	0.43133635377821	0.67223037060286	0.8199162846655
IC mínimo	0.10988123752695	0.24827523514815	0.42358918316877	0.66371826005441	0.81443443994121

Tabela 10: Esperança W1 esperado x simulado

ρ	0.2	0.4	0.6	0.8	0.9
E[T1] esperado	1.111111111	1.25	1.428571428	1.666666667	1.8181818182
E[T1] simulado	1.11023254250302	1.25021866640229	1.42692638214542	1.66757154267597	1.81716278011947
IC máximo	1.11270810449139	1.25320710166536	1.43200572231137	1.6722867741044	1.82029417447625
IC mínimo	1.10775698051464	1.24723023113922	1.42184704197947	1.66285631124753	1.8140313857627

Tabela 11: Esperança T1 esperado x simulado

ρ	0.2	0.4	0.6	0.8	0.9
E[W2] esperado	0.527777778	1.5	3.6428571428	10.66666667	25.363636364
E[W2] simulado	0.52597275104699	1.50144022352571	3.61733881908095	10.6745457878966	25.2799700922782
IC máximo	0.52960047200885	1.51470834948598	3.65100056239466	10.816718996599	25.5654287666901
IC mínimo	0.52234503008513	1.48817209756544	3.58367707576723	10.5323725791942	24.9945114178663

Tabela 12: Esperança W2 esperado x simulado

ρ	0.2	0.4	0.6	0.8	0.9
E[T2] esperado	1.527777778	2.5	4.6428571428	11.66666667	26.363636364
E[T2] simulado	1.5255100040633	2.50103523526766	4.61780669830138	11.6742186334784	26.2800652018098
IC máximo	1.5304346134045	2.51497583377976	4.65176274027073	11.8168127976126	26.565792938344
IC mínimo	1.52058539472211	2.48709463675556	4.58385065633203	11.5316244693442	25.9943374652756

Tabela 13: Esperança T2 esperado x simulado

ρ	0.2	0.4	0.6	0.8	0.9
E[Nq1] esperado	0.0111111111	0.05	0.128571428	0.266666667	0.3681818182
E[Nq1] simulado	0.01100799102969	0.0500587897651	0.12808867195671	0.26723775639467	0.36775387729734
IC máximo	0.01113722051083	0.050477711357432	0.12928638748629	0.26914182756829	0.3690461140906
IC mínimo	0.010878761548549	0.049639868172767	0.12689095642713	0.26533368522106	0.36646164050408

Tabela 14: Esperança Nq1 esperado x simulado

ρ	0.2	0.4	0.6	0.8	0.9
E[Nq2] esperado	0.052777778	0.3	1.092857142	4.266666667	11.413636364
E[Nq2] simulado	0.052469606088934	0.3004273909326	1.08400363176704	4.2708480816448	11.3770691373038
IC máximo	0.052878487927044	0.3033460296362	1.09494568094651	4.33075721326658	11.5082143124492
IC mínimo	0.052060724250824	0.29750875222901	1.07306158258756	4.21093895002301	11.2459239621583

Tabela 15: Esperança Nq2 esperado x simulado

ρ	0.2	0.4	0.6	0.8	0.9
E[N1] esperado	0.111111111	0.25	0.428571428	0.666666667	0.8181818182
E[N1] simulado	0.11074863557422	0.25014245198777	0.42757213936586	0.66712860751545	0.81777623631782
IC máximo	0.11110879955635	0.25098003236157	0.42930185751079	0.66956348617731	0.81932513062072
IC mínimo	0.11038847159209	0.24930487161397	0.42584242122092	0.66469372885358	0.81622734201492

Tabela 16: CoEsperança N1 esperado x simulado

ρ	0.2	0.4	0.6	0.8	0.9
E[N2] esperado	0.152777778	0.5	1.3928571428	4.666666667	11.863636364
E[N2] simulado	0.15217623593395	0.50042323488799	1.38378695423375	4.67076649463562	11.8271401332377
IC máximo	0.15283461847176	0.50368153012956	1.39508497581679	4.73112982696196	11.9585143917964
IC mínimo	0.15151785339615	0.49716493964643	1.3724889326507	4.61040316230929	11.695765874679

Tabela 17: Esperança N2 esperado x simulado

ρ	0.2	0.4	0.6	0.8	0.9
V(W1) simulado	0.23299658280757	0.56038914786681	1.04109211540801	1.78426804154869	2.29646702598462
IC máximo	0.23782124945894	0.56779833167504	1.06175302535004	1.81112190853202	2.31301770374322
IC mínimo	0.2281719161562	0.55297996405857	1.02043120546597	1.75741417456537	2.27991634822602

Tabela 18: variância de W1 simulado

ρ	0.2	0.4	0.6	0.8	0.9
V(W2) simulado	1.36902569781281	5.65490050212992	22.3689706500987	148.781613892439	732.165757236055
IC máximo	1.39162431444299	5.78046918411458	23.0250308123053	154.726860023532	760.130153166632
IC mínimo	1.34642708118263	5.52933182014527	21.7129104878921	142.836367761346	704.201361305478

Tabela 19: variância de W2 simulado

- **Gráfico de E[W1]**

ρ	Esperado	Encontrado	Maximo	Minimo
0.2	0.11111111111111	0.11219425645882	0.11341654539293	0.11097196752471
0.4	0.25	0.25006381104239	0.25191218210573	0.24821543997905
0.6	0.42857142857143	0.42911541549024	0.43222177638715	0.42600905459332
0.8	0.66666666666667	0.66797431532864	0.67223037060286	0.66371826005441
0.9	0.81818181818182	0.81717536230335	0.8199162846655	0.81443443994121

Tabela 20: valores de E[W1]

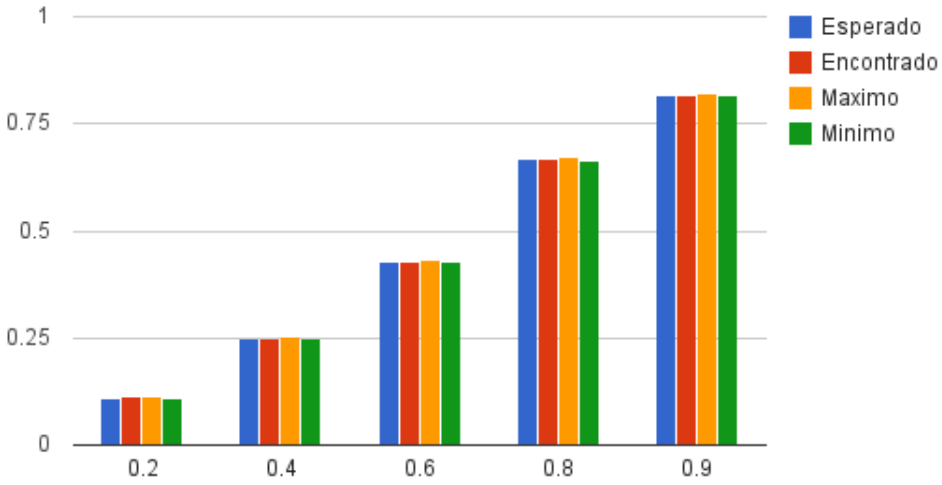

Figura 3: Gráfico de E[W1]

- **Gráfico de E[Nq1]**

ρ	Esperado	Encontrado	Maximo	Minimo
0.2	0.011111111111111	0.011209642970553	0.011338986172909	0.011080299768198
0.4	0.05	0.050008211402201	0.050413364777661	0.04960305802674
0.6	0.12857142857143	0.12877350638152	0.12981185823908	0.12773515452396
0.8	0.26666666666667	0.26723775639467	0.26914182756829	0.26533368522106
0.9	0.36818181818182	0.36775387729734	0.3690461140906	0.36646164050408

Tabela 21: valores de E[Nq1]

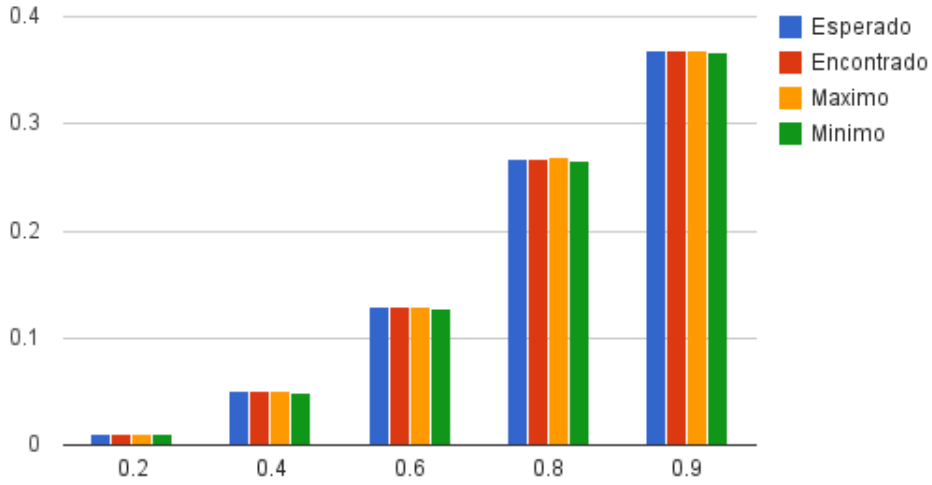

Figura 4: Gráfico de E[Nq1]

- **Gráfico de E[T1]**

ρ	Esperado	Encontrado	Maximo	Minimo
0.2	1.11111111111111	1.11351613947233	1.11584068356874	1.11119159537592
0.4	1.25	1.24873086883092	1.25190036009894	1.2455613775629
0.6	1.42857142857143	1.42820652048646	1.43243881748345	1.42397422348947
0.8	1.66666666666667	1.66757154267597	1.6722867741044	1.66285631124753
0.9	1.81818181818182	1.81716278011947	1.82029417447625	1.8140313857627

Tabela 22: valores de E[T1]

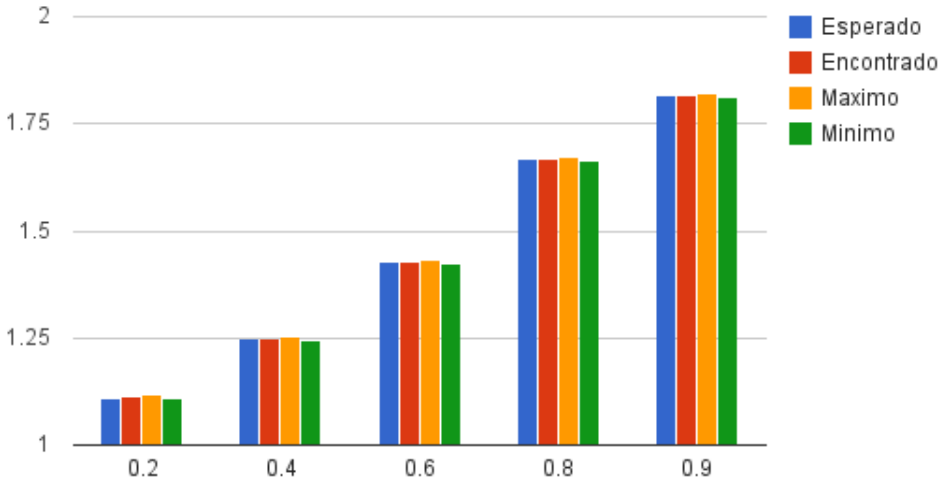

Figura 5: Gráfico de E[T1]

- **Gráfico de E[N1]**

ρ	Esperado	Encontrado	Maximo	Minimo
0.2	0.11111111111111	0.11124715990523	0.11156826408472	0.11092605572575
0.4	0.25	0.24971283790112	0.25053203271413	0.2488936430881
0.6	0.42857142857143	0.42857279925318	0.43021549392611	0.42693010458026
0.8	0.66666666666667	0.66712860751545	0.66956348617731	0.66469372885358
0.9	0.81818181818182	0.81777623631782	0.81932513062072	0.81622734201492

Tabela 23: valores de E[N1]

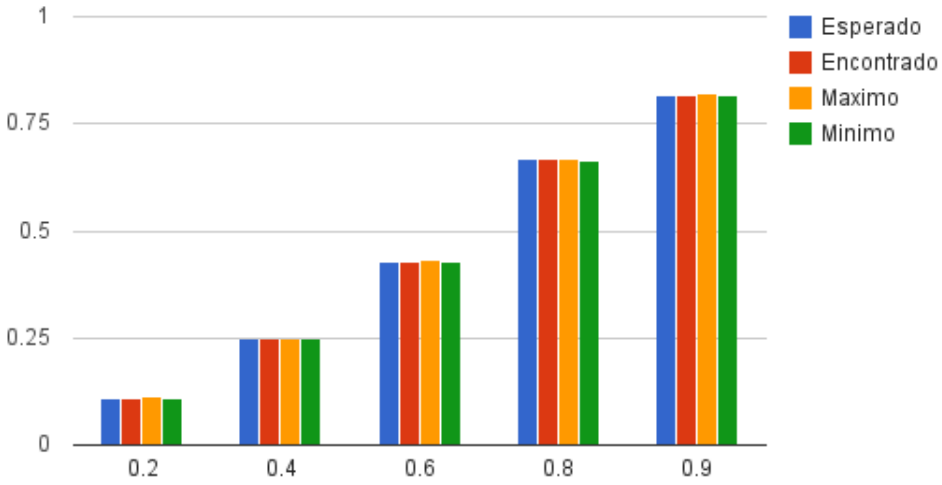

Figura 6: Gráfico de E[N1]

- **Gráfico de E[W2]**

ρ	Esperado	Encontrado	Maximo	Minimo
0.2	0.52777777777778	0.52872616965278	0.53281299637416	0.52463934293141
0.4	1.5	1.50022751178363	1.51436135926444	1.48609366430282
0.6	3.64285714285714	3.63072742608245	3.66827410034753	3.59318075181737
0.8	10.6666666666667	10.6745457878966	10.816718996599	10.5323725791942
0.9	25.3636363636364	25.2799700922782	25.5654287666901	24.9945114178663

Tabela 24: valores de E[W2]

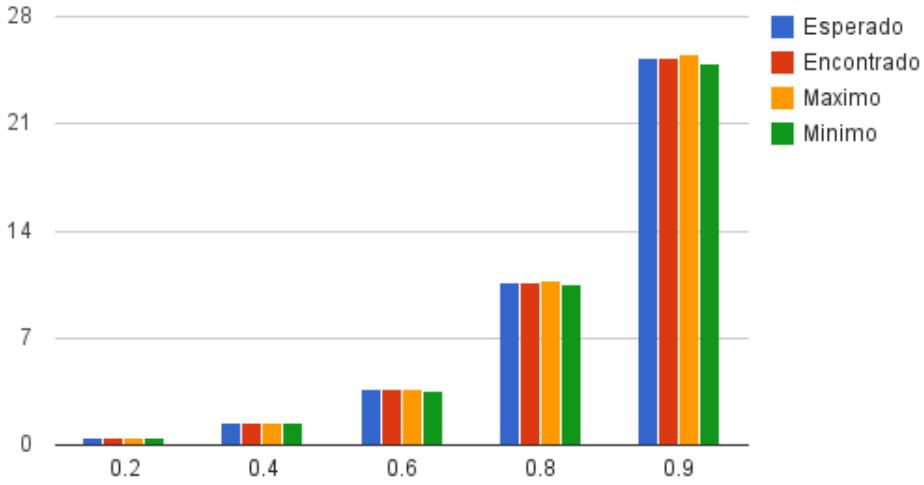

Figura 7: Gráfico de E[W2]

- **Gráfico de E[Nq2]**

ρ	Esperado	Encontrado	Maximo	Minimo
0.2	0.052777777777778	0.052824522969304	0.053261358014342	0.052387687924266
0.4	0.3	0.30003697635306	0.30318335060684	0.29689060209928
0.6	1.09285714285714	1.08960654437444	1.10194242488134	1.07727066386753
0.8	4.26666666666667	4.2708480816448	4.33075721326658	4.21093895002301
0.9	11.4136363636364	11.3770691373038	11.5082143124492	11.2459239621583

Tabela 25: valores de E[Nq2]

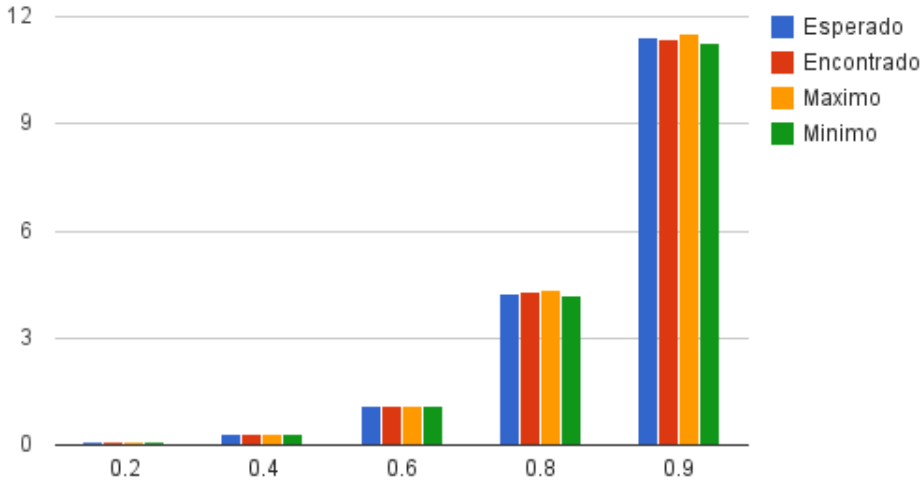

Figura 8: Gráfico de E[Nq2]

- **Gráfico de E[T2]**

ρ	Esperado	Encontrado	Maximo	Minimo
0.2	1.52777777777778	1.52967803675752	1.53461548194256	1.52474059157249
0.4	2.5	2.50038721593718	2.51503307611335	2.48574135576101
0.6	4.64285714285714	4.62946384570244	4.66733109497981	4.59159659642506
0.8	11.6666666666667	11.6742186334784	11.8168127976126	11.5316244693442
0.9	26.3636363636364	26.2800652018098	26.565792938344	25.9943374652756

Tabela 26: valores de E[T2]

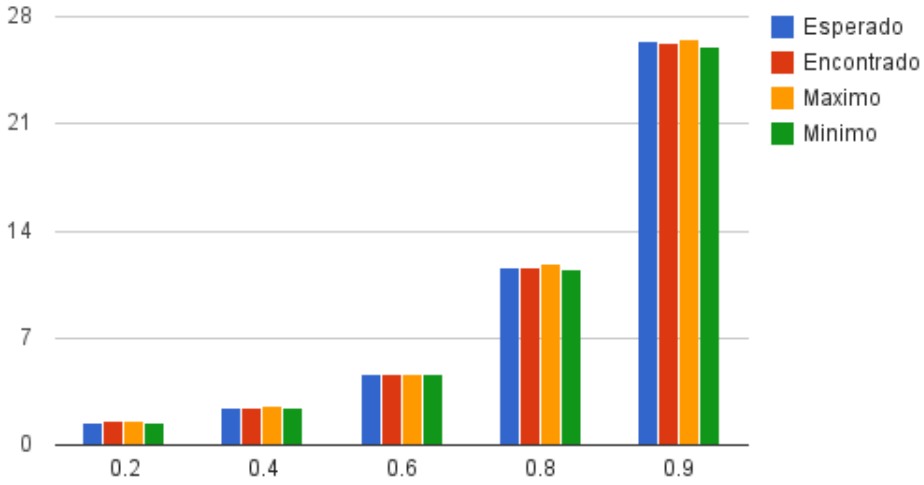

Figura 9: Gráfico de E[T2]

- **Gráfico de E[N2]**

ρ	Esperado	Encontrado	Maximo	Minimo
0.2	0.15277777777778	0.15282537790103	0.15342712044192	0.15222363536013
0.4	0.5	0.50004367176965	0.50354770117085	0.49653964236844
0.6	1.39285714285714	1.38930100881724	1.40206438039037	1.37653763724412
0.8	4.66666666666667	4.67076649463562	4.73112982696196	4.61040316230929
0.9	11.8636363636364	11.8271401332377	11.9585143917964	11.695765874679

Tabela 27: valores de E[N2]

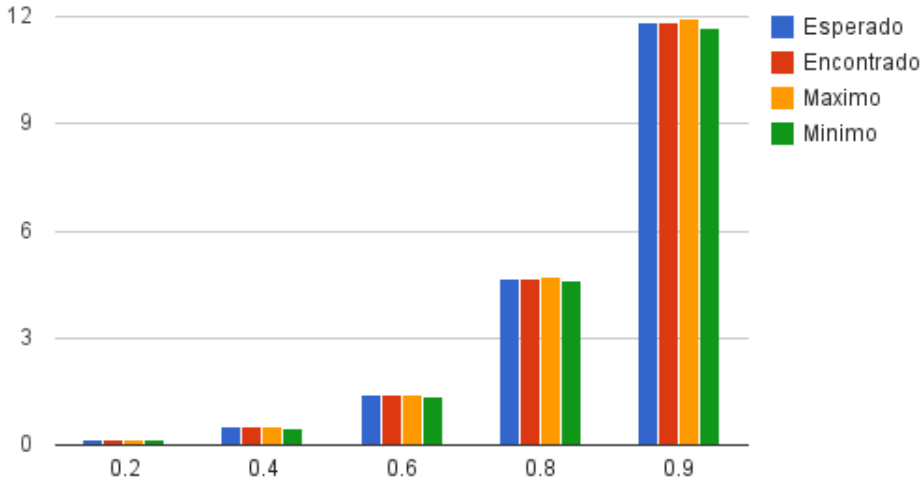

Figura 10: Gráfico de E[N2]

2.5 Otimização

2.6 Conclusão

2.7 Anexo

Foi gerado como parte da documentação do programa o Javadoc, é um gerador de documentação criado pela Sun Microsystems para documentar a API dos programas em Java, a partir do código-fonte. O resultado é expresso em HTML. É constituído, basicamente, por algumas marcações muitos simples inseridas nos comentários do programa.

Estrutura interna da pasta Trabalho_Final_MAB515 , e um explicação do que econtramos nela:

doc	Pasta onde está localizado o Javadoc.
Relatório	Onde se econtra o relatório do programa.
Simulador	Nela encontramos dentro da pasta src o código.
Simulador.jar	arquivo que contém todas as classes compactadas , para poder executar o simulador.
simulador.bat	arquivo que servirá para executar o comando para inicializar o servidor.Funciona apenas no windows.
simulador.sh	arquivo que servirá para executar o comando para inicializar o servidor.Funciona apenas no Linux.
XML	Onde se econtra o config.xml para quando rodar o simulador.jar, ele poder ler este arquivo.

Segue abaixo uma explicação superfícial do que as classes e seus metódos fazem, para obter melhores informações sobre os métodos, como qual são seus parâmetros o que retornam vá ao javadoc que lá irá encontrar tudo mais explicado. O programa está dividido em quatro pacotes.

- br.ufrj.dcc.controle
  - **GeradorAleatorio.java**
    Classe que gera as amostras do sistema.
    - *GeradorAleatorio*
      Construtor da classe neste caso é private, porque implementamos o padrão de projeto singleton, que faz com que haja apenas uma instância deste objeto no sistema.
    - *getInstance*
      Retorna a instância desta classe, quando este método é chamado pela primeira vez será criada a mesma.
    - *getGeraAmostra*
      Gera uma amostra usando a fórmula $x_0 = -\frac{\ln u_0}{taxa}$, onde a taxa é passada como parâmetro.
  - **GerenciadorEventos.java**
    Classe que gerência os eventos do programa que podem ser chegada ou sada ou interrupção.
    - *geraEventoSaida*
      Método que gera um evento de saida do sistema ou seja pedo um freguês de uma das duas filas e coloco em serviço.
    - *geraEventoChegada*
      Gera um evento de chegada ou seja um cliente chegando a fila 1.
    - *geraEventoDeInterrupcao*
      Sempre é chamado quando ocorre um evento de chegada e quem está em serviço é um freguês oriundo fa fila2 , com isso tiramos

o cliente em serviço e devolvemos a sua fila e pegamos o novo cliente em e colocamos no servidor.

  * *pegarProximoEvento*
    Pega o próximo evento a ser tratado de acordo com sua hora de ocorrência,os eventos podem ser de chegada ou saida.
  * *getProximoEventoSaida*
    Retorna o próximo evento de saida.

- **ManipulaXML.java**
  Classe que realiza a leitura do config.xml
  * *ManipulaXML*
    Construtor da classe que recebe como parâmetro o caminho do arquivo , como padrão está o diretório do projeto na pasta XML.
  * *leConfiguracao*
    Abre o arquivo e começa a ler as tags.
  * *getChildTagValue*
    Recebe como parâmetro o nome da tag e retorna o seu valor.
- **Rodada.java**
  Classe que realiza a rodada
  * *Rodada*
    Construtor da Classe recebe como parâmetro um objeto da classe de Configuração e Resultado, para poder verficar os parâmetros para cada rodada e amarzenar o resultado de cada rodada.
  * *simulacao*
    Realiza efetivamente a rodada nela crio as filas de acordo com disciplina de atendimento informada, a partir dai começa-se a gerar chagadas no sistema.
- **Simulador.java**
  Classe que é chamada ao iniciar o sistema.
  * *main*
    Primeiro método a sere chamado assim que o programa é iniciado , nele criamos um "loop"de acordo com a quantidade de rodadas que serão feitas para após retornar os resultados estatísticos.
  * *calculaIntervaloDeConfianca*
    Realiza o cálculo do intervalo de confiança fazendo uso da T-Student como o valor de 1.96 , o que nos dá uma precisão de 95%.
  * *calculaMedia*
    Realiza o cálculo da média para exbir a reposta , fazendo uso do valor médio obtido em cada rodada.

• br.ufrj.dcc.modelo

- **Configuaracao.java**
  Armazena os parâmetros que serão utilizados na simulação.
  * *Configuracao*
    Construtor da classe recebe como parâmetro os seguintes valores: fase transiente, número de rodadas,tamanho das rodadas,taxa de serviço , utilização e a disciplina de atendimento das duas filas.

* *getFaseTransiente*
  Retorna a quatidade de clientes que pertecem a fase transiente.
* *getNumeroRodada*
  Retorna o número de rodadas que iermos fazer.
* *getTamanhoRodada*
  Retorna a quantidade de eventos de chegada que iremos gerar.
* *getTaxaServico*
  Retorna a taxa de serviço do servidor.
* *getUtilizacao*
  Retorna a utilização do sistema.
* *getFila1*
  Retorna a disciplina de atendimento da fila1
* *getFila2*

– **Evento.java**
Classe que armazena os valores especificados no xml.

* *Evento*
  Construtor nesta classe temos dois ,quando passamos como parâmetro um frêgues indica que estamos criando um evento de chegada caso seja o servidor estamos criando um evento de Saida.
* *isEventoChegada*
  Retorna true se o evento é de chegada .
* *isEventoSaida*
  Retorna true se o evento é de saida.
* *getFregues*
  Retorna o freguês que gerou o evento , apenas utilizado se for de chegada.
* *getTipo*
  Retorna o tipo do evento que pode ser de chegada ou saida.
* *getHoraOcorrencia*
  Retorna a hora que irá ocorrer o evento.
* *getServer*
  Retorna o servidor ao qual iremos colocar o freguês, apenas utilizado para evento de saida.

– **Fregues.java**
Classe que faz o papel de um freguês no sistema.

* *setCor*
  indentifica se o cliente pertence a fase transiente.
* *getCor*
  Retorna o valor do campo que unicamente pode ser preenchido com "TRANSIENTE"ou "NAOTRANSIENTE".
* *setInstanteChegadaFila1*
  Preenche a variável que indica quando o cliente chegou no sistema e entrou na fila1.
* *getInstanteChegadaFila1*
  Retorna o valor da variável.

* *setInstanteSaidaFila1*
  Preenche a variável que indica quando o cliente saiu da fila1 e foi para o servidor.
* *getInstanteSaidaFila1*
  Retorna o valor da variável.
* *setInstanteChegadaFila2*
  Preenche a variável que indica quando o instante de chegada do cliente a fila 2.
* *getInstanteChegadaFila2*
  Retorna o valor da variável.
* *setInstanteSaidaFila2*
  Preenche a variável que indica momento em que saiu da fila2 e foi para o servidor.
* *getInstanteSaidaFila2*
  Retorna o valor da variável.
* *setInstanteSaidaSistema*
  Preenche a variável que indica momento em que acabou de ser servido e pode ir embora do sistema
* *getInstanteSaidaSistema*
  Retorna o valor da variável.
* *setServicoResidual*
  Preenche a variável que indica caso o tempo falta para acabar de ser servido.Este campo é preenchido quando ocorre uma interrupção.
* *getServicoResidual*
  Retorna o valor da variável.
* *setFilaOrigem*
  Preenche a variável que indica para o que faremos com o frêgues após ser servido se pode ir embora ou o colocamos na fila2.
* *getFilaOrigem*
  Retorna o valor da variável.
* *chegarNoSistema*
  Preencha as variáveis necessárias como instante de chegada na fila, e inseri o novo

– **Resultado.java**
Classe que armazena o resultado de cara Rodada.

* *getT1*
  Retorna um "array"com tempo total médio de que um freguês levou para sair do sistema 1(sair da fila mais acabar ser ser servido), de cada rodada.
* *getT2*
  Retorna um "array"com tempo total médio de que um freguês levou para sair do sistema 2(sair da fila mais acabar ser ser servido), de cada rodada.
* *getW1*
  Retorna um "array"com tempo de espera médio de um cliente na fila1, de cada rodada.

* *getW2*
  Retorna um "array"com tempo de espera médio de um cliente na fila2, de cada rodada.
* *getN1*
  Retorna um "array"com número médio de clientes que passaram pela fila1, de cada rodada.
* *getN2*
  Retorna um "array"com número médio de clientes que passaram pela fila2, de cada rodada.
* *getNq1*
  Retorna um "array"com o número médio de fregueses que ficaram esperando na fila 1, de cada rodada.
* *getNq2*
  Retorna um "array"com o número médio de fregueses que ficaram esperando na fila 2, de cada rodada.
* *getVarianciaW1*
  Retorna um "array"com a variância do tempo de espera médio na fila1 em acada rodada.
* *getVarianciaW2*
  Retorna um "array"com a variância do tempo de espera médio na fila2 em acada rodada.
* *setT1*
  adiciona ao "array"o tempo total médio de que um freguês levou para sair do sistema 1(sair da fila mais acabar ser ser servido) na última rodada.
* *setT2*
  adiciona ao "array"o tempo total médio de que um freguês levou para sair do sistema 2(sair da fila mais acabar ser ser servido) na última rodada.
* *setW1*
  adiciona ao "array"o tempo de espera médio de um cliente na fila1 na última rodada.
* *setW2*
  adiciona ao "array"o tempo de espera médio de um cliente na fila2 na última rodada.
* *setN1*
  adiciona ao "array"o número médio de clientes que passaram pela fila1 na última rodada.
* *setN2*
  adiciona ao "array"o número médio de clientes que passaram pela fila2 na última rodada.
* *setNq1*
  adiciona ao "array"o número médio de fregueses que ficaram esperando na fila 1 na última rodada.
* *setNq2*
  adiciona ao "array"o número médio de fregueses que ficaram esperando na fila 2 na última rodada.

    - *setVariancia W1*
      adiciona ao "array"a variância do tempo de espera médio na fila1 na última rodada.
    - *setVariancia W2*
      adiciona ao "array"a variância do tempo de espera médio na fila2 na última rodada.
  - **Servidor.java**
    Classe que faz o papel do servidor do sistema.
    - *tempoServico*
      Manda o gerador aleatório gerar uma amostrar que indica quanto tempo o cliente levará em serviço.
    - *isOcioso*
      Retorna true se não tem niguém em serviço.
    - *getFregues*
      Retorna o freguês que está em serviço.
    - *getFilaOrigem*
      Retorna a fila de origem do cliente que está em serviço.
    - *finish*
      Método que verifica se tem alguém para ser servido na fila1 ou fila2 e realiza o termino do serviço do cliente atual.
    - *servicoCompleto*
      Este metódo decide o que faço com o cliente que acabou de ser servido se coloco na fila 2 ou se pode ir embora do sistema.
    - *trataInterrupcao*
      Método que trata interrupção ou seja coloca o novo cliente em serviço , coloca o antigo na sua fila e preenche o tempo de serviço residual.
    - *geraResultado*
      gera o resultado da última rodada.
    - *getResultado*
      retorna o resultado da última rodada.

- br.ufrj.dcc.modelo.enumerator
  - **CorCliente.java**
    enumerator criado para representar as possíveis cores que um cliente pode assumir, como no nosso caso estamos usando o método replicativo só podem ser duas "TRANSIENTE"ou "NAOTRANSIENTE".
  - **FilaOrigem.java**
    enumerator criado para localizar aonde os freguês se encontra no sistema:"FILA1"e "FILA2".
  - **TipoEvento.java**
    enumerator criado para destacar apenas os eventos gerados no sistema que são: "SAIDA"e "CHEGADA".

- br.ufrj.dcc.modelo.fila
  - **Fila.java**
    É uma superclasse da FCFS e LCFS, utilizamos ela para não precisar

implementar todos os metódos duas vezes apenas os que são direntes em cada disciplina de atendimento.

  * *Fila*
    Está classe também possui dois construtors, um para a fila 1 que recebe como parâmetro o servidor, λ que é taxa com que os freguese chegam ao sistema como também a fila 2, a qual tenho mais prioridada e outro para a fila2 pois nela não ocorrem chegadas e nem é mais prioritária que outra fila , dessa forma só recebe o serviço.
  * *removeParaAtendimento*
    deve ser implementado pela classe que herda dessa.
  * *addFregues*
    deve ser implementado pela classe que herda dessa.
  * *addFreguesResidual*
    deve ser implementado pela classe que herda dessa.
  * *calculaProximaChegada*
    Manda o gerador aleatório gerar uma amostrar que indica quando irá chegar um cliente no sistema.
  * *isEmpty*
    Retorna true se a fila está vazia.
  * *size*
    Retorna o tamanho da fila no istante que este metódo é chamado.
  * *getNumeroTotalChegadas*
    Retorna a quantidade de pessoas que já passaram pela fila.
  * *getLambda*
    Retorna a taxa de chegada, que no caso da fila1 será o valor de λ, já na fila2 será 0.0

- **FCFS.java**

  * *FCFS*
    Crio objeto do tipo FCFS e recebe os parâmetros e passa para a super-classe.
  * *removeParaAtendimento*
    Retorna o primeiro freguês da fila para ser atendido.
  * *addFregues*
    Adiciona o cliente que acabou de chegar ao final da fila.
  * *addFreguesResidual*
    Inseri o cliente que possui resíduo ou seja que sofreu um interrupção no íncio da fila.

- **LCFS.java**

  * *LCFS*
    Crio objeto do tipo FCFS e recebe os parâmetros e passa para a super-classe.
  * *removeParaAtendimento*
    Retornat o primeiro freguês da pilha para ser atendido

* *addFregues*
  Adiciona o cliente que acabou de chegar no inicio da fila.
* *addFreguesResidual*
  Inseri o cliente que possui resíduo ou seja que sofreu um interrupção no íncio da fila.