UNIVERSIDADE FEDERAL DA PARAÍBA - UFPB
CENTRO DE CIÊNCIAS E TECNOLOGIA - CCT
CURSO DE MESTRADO EM INFORMÁTICA

SIM/SAVAD

UM SIMULADOR DE MODELOS DE REDES DE FILAS

Haroldo Castro Conceição Filho

CAMPINA GRANDE

Dezembro - 1993

SIM/SAVAD - UM SIMULADOR DE MODELOS DE REDES DE FILAS

Haroldo Castro Conceição Filho

Dissertação apresentada ao curso de MESTRADO EM INFORMÁTICA da UNIVERSIDADE FEDERAL DA PARAÍBA, em cumprimento às exigências para obtenção do grau de mestre.

Área de Concentração: Ciência da Computação

MARIA IZABEL CAVALCANTI CABRAL
Orientadora

Campina Grande - PB

Dezembro - 1993

C744s Conceição Filho, Haroldo Castro.
SIM/SAVAD : um simulador de modelos de redes de filas / Haroldo Castro Conceição Filho. - Campina Grande, 1993.
162 f.

Dissertação (Mestrado em Informática) - Universidade Federal da Paraíba, Centro de Ciências e Tecnologia, 1993.
"Orientação : Profa. Dra. Maria Izabel Cavalcanti Cabral".
Referências.

1. Redes de Computadores. 2. Redes de Filas. 3. SIM/SAVAD. 4. Dissertação - Informática. I. Cabral, Maria Izabel Cavalcanti. II. Universidade Federal da Paraíba - Campina Grande (PB). III. Título

CDU 004.7(043)

SIM/SAVAD-UM SIMULADOR DE MODELOS DE REDES DE FILAS

HAROLDO CASTRO CONCEIÇAO FILHO

DISSERTAÇAO APROVADA EM 21.12.1993

MARIA IZABEL CAVALCANTI CABRAL, D.Sc
Orientador

MARCOS ANTONIO GONÇALVES BRASILEIRO, Dr.
Componente da Banca

CLAUDINE CHAOUIYA, Dra.
Componente da Banca

Campina Grande, 21 de dezembro de 1993

AGRADECIMENTOS

A Deus por me dar o caminho.

A minha orientadora, Profa. Dra. Maria Izabel Cavalcanti Cabral, pelo apoio e dedicação a mim dispensados.

Aos professores e funcionários.

Aos meus familiares, amigos e colegas que contribuíram direta ou indiretamente para este trabalho.

RESUMO

Esta Dissertação apresenta o SIM/SAVAD, um simulador de modelos de redes de filas que pode representar uma grande variedade de sistemas que exibem contenção de recursos. São exemplos: sistemas de computadores, de redes de computadores, de manufatura e de tráfego.

Os elementos de modelagem do SIM/SAVAD permitem facilmente modelar sistemas de redes de filas, particularmente sistemas de redes de computadores.

O uso do paradigma de objetos na construção do SIM/SAVAD viabilizou um projeto modular, extensível e reutilizável, como também, a exploração dos mecanismos de herança e amarração dinâmica, suportados pela linguagem de programação C++, usada na implementação do SIM/SAVAD.

O SIM/SAVAD também é parte integrante do módulo solução do Sistema de Avaliação de Desempenho (SAVAD), um projeto do Grupo de Redes de Computadores e do Grupo de Inteligência Artificial da UFPB (GRC/GIA).

ABSTRACT

This dissertation presents SIM/SAVAD, a simulator of models for networks of queues. The models can represent a large variety of systems exhibiting resource contention, as, for example: computer systems, computer networks systems, manufacturing systems, and trafic-control systems.

SIM/SAVAD´s modelling elements allow a non expert to easily model systems of network of queues, particularly computer network systems.

The use of the object-oriented paradigm in both the project and the implementation of SIM/SAVAD favored its modularity, extensibility, and reusability, as well as providing the advantages of the mechanisms of inheritance and of dynamic binding supported by the adopted programming language, C++.

SIM/SAVAD composes part of the solution module of SAVAD, a Performance Evaluation System, currently being developed by the Computer Network Research Group (GRC) and the Artificial Intelligence Research Group (GIA) of the Federal University of Paraíba (UFPB).

SUMÁRIO

FIGURAS

TABELAS

CAPITULO 1

INTRODUCAO

C A P Í T U L O 1

INTRODUÇÃO

Avaliar o desempenho de um sistema é avaliar a capacidade deste sistema no atendimento de sua demanda de serviço. Para essa avaliação definem-se medidas de desempenho de interesse e observam-se os comportamentos dessas medidas diante de diferentes situações possíveis para o sistema em estudo.

Por sua vez, a observação do comportamento de um sistema (estudo do seu desempenho) pode ser feito através de **medições** ou de **modelagem**. A modelagem geralmente é usada quando o estudo do sistema em si (medições) não é possível ou então não é uma opção viável ou a mais adequada.

Na modelagem são utilizadas formas de representação que permitem manipular e compreender as entidades estudadas em seus aspectos qualitativos e quantitativos. Estas formas de representação de sistemas são denominadas **modelos**.

Os sistemas que apresentam contenção de recursos são geralmente modelados usando o **paradigma de redes de filas** [Kreutzer 86:38]. Sistemas de computação, redes de computadores e sistemas de tráfego são exemplos de sistemas com contenção de recursos.

Solucionar modelos de redes de filas ainda é uma tarefa complexa, executada apenas por aqueles que possuem amplo conhecimento das técnicas disponíveis na literatura especializada.

Para solucionar modelos de redes de filas podem-se utilizar **técnicas exatas** (analíticas) e **técnicas aproximadas**.

As técnicas analíticas geralmente baseiam-se na Teoria das Filas [Kleinrock 75]. A abordagem analítica é usualmente mais econômica e eficiente quando aplicável, embora, muitas vezes, exija suposições simplificadoras que inviabilizam a sua utilização. [Sauer 81] apresenta as características de modelos de redes de filas que inviabilizam a escolha de uma solução analítica.

Dentre as técnicas aproximadas destaca-se a **Simulação Digital** (abrangente para qualquer tipo de redes de filas: fechadas, abertas e mistas). A Simulação Digital pode ser definida como um experimento estatístico que observa o comportamento do modelo e obtem medidas de desempenho a partir desta observação.

Segue um quadro comparativo entre a solução analítica e a solução usando a técnica da Simulação Digital [Cabral 90]:

Solução Analítica	Simulação Digital
Apresenta relações das caracteristicas do modelo e seus efeitos no desempenho do modelo	Camuflagem das relações entre causas e efeitos
Econômica Eficiente	Alto custo computacional para se obterem níveis de intervalo de confiança desejáveis
Modelos simplificados	Aplicável a qualquer modelo

Segundo [Kreutzer 86:6], a grande atração que a técnica da simulação possui é a sua aplicabilidade. Além disto, frequentemente, não existe outra técnica para construir e explorar modelos de sistemas complexos, irregulares e com alta interconectividade.

A simulação utiliza modelos que possuem uma estrutura matemática/lógica que pode ser exercitada de forma a mimetizar o comportamento do sistema. Através de simulação várias observações são realizadas para dar subsídio as várias conclusões sobre o sistema [Soares 90:3].

No caso de modelos que envolvem muitas operações e um grande número de variáveis, um computador pode ser utilizado para exercitar o modelo. Neste caso o modelo deve ser compreensível pelo computador. Neste sentido, um conceito chave é o da descrição dos estados do sistema.

Descrever estados de um sistema consiste na descrição dos domínios dos valores assumidos pelos atributos (características) dos componentes do sistema.

"Se um sistema pode ser descrito por um conjunto de **variáveis**, cuja uma combinação de valores represente um único estado ou condição do sistema, então a manipulação dos valores das variáveis simula o movimento de estado a outro" [Soares 90:4].

A forma como ocorrem (e consequentemente são modeladas) as mudanças de estados de um sistema podem ser **contínuas** ou **discretas** no tempo. A simulação contínua modela sistemas nos quais a troca de estados do modelo ocorre continuamente no tempo (ex. modelos usando equações diferenciais).

Na forma discreta a mudança de estados ocorre em instantes específicos do tempo, que podem ser estabelecidos de forma **determinística** ou **aleatória.**

Os processos que ocorrem de forma aleatória devem ser estudados para encontrar as **distribuições estatísticas** que os representem. São exemplos de processos aleatórios: processo de interchegada de clientes e processo de atendimento de serviço dos clientes).

Quanto aos resultados deve-se observar: "..., as saídas produzidas pelo sistema devem ser **inferidas** a partir das saídas do modelo. Uma correspondência direta entre as saídas do modelo e do sistema pode não ser possível devido às simplificações feitas quando da modelagem. É parcialmente por esta razão que a simulação num modelo é frequentemente usada para determinar **tendências** no comportamento do sistema modelado ao invés de obter valores reais para as suas saídas. A inferência também é necessária desde que uma simulação é um experimento **estocástico** cujos resultados têm ocorrências que devem ser associadas a intervalos de confiança" [Moura 86:280].

Um programa para simulação (simulador) pode ser feito de duas formas:

a) usando ferramentas gerais e poderosas (linguagens de simulação (ex. GPSS) ou pacotes baseados em sub-rotinas escritas em linguagens de programação de propósito geral (ex . GASP)),

b) usando linguagens de programação de propósito geral (no caso desta dissertação, C++ [Dewrust 90], uma linguagem orientada a objetos).

A forma (b) é indicada para tirar proveito da simplicidade ou peculiaridade do modelo, demonstrar claramente a lógica necessária numa simulação, esclarecer características importantes existentes nas ferramentas da forma (a) e, conforme [Moura 86:282], possibilita-nos expor os conceitos de simulação de forma mais objetiva, sem apartes distrativos para explicação dos detalhes das ferramentas ou pacotes utilizados para definir simuladores.

Programas para simulação são peças complexas de software, as quais devem ser construídas de forma bem disciplinada, bem estruturadas e de uma forma segura. [Kreutzer 86:12] menciona que entre as metodologias de programação comumente usadas, a **programação orientada a objetos** exibe o melhor potencial para descrever sistemas complexos de uma maneira natural e bem estruturada.

A programação orientada a objetos envolve o estudo de técnicas, notações e ferramentas que estudam como disciplinar o processo de abstração para a construção de sistemas com características estruturais e dinâmicas bem definidas. A base desta metodologia parte da assertiva de que o mundo real é composto por objetos e não por dados ou processos, devendo o desenvolvimento de software estar baseado na identificação e manipulação de objetos. Linguagens de programação orientada a objetos dispõem de mecanismos tais como herança e amarração dinâmica que viabilizam a reutilização de software [Takahashi 90].

1.1 - Objetivo

Essa dissertação objetiva a construção de um simulador de modelos de redes de filas, denominado SIM/SAVAD. Esses modelos, que apresentam contenção de recursos, podem representar sistemas diversos. São exemplos: sistemas de computadores, de redes de computadores, de manufatura e de tráfego (aeroportos, ferrovias, etc.).

O SIM/SAVAD, construído usando o paradigma de objetos, apresenta facilidades que tornam mais abrangente a sua utilização na simulação de modelos de redes de filas. Entre elas, destacam-se:

1) Um conjunto de elementos que facilitam a construção de modelos de redes de filas, particularmente, no que se refere a modelos de redes de computadores.

2) Uma interface amigável que permite aos usuários:

a) construir e validar modelos de redes de filas,

b) escolher níveis de confiança para as medidas de desempenho e,

c) apresentar os resultados da simulação.

3) Um projeto modular, extensível e reutilizável, propiciado pelas potencialidades da abordagem orientada a objetos exploradas nesse projeto.

4) Uso de mecanismos tais como herança e amarração dinâmica, suportados pela linguagem de programação C++, usada na implementação do SIM/SAVAD.

O SIM/SAVAD também é parte integrante do módulo solução do Sistema de Avaliação de Desempenho (SAVAD), um projeto em desenvolvimento pelo Grupo de Redes de Computadores e Grupo de Inteligência Artificial da UFPB (GRC/GIA).

O SAVAD é um exemplo de ambiente de simulação inteligente, ele dispõe de uma interface amigável e de um sistema especialista. A interface recebe especificações de modelos de redes de filas, enquanto o sistema especialista procura, automaticamente, a melhor forma de solucioná-los, ativando um dos módulos de solução do SAVAD.

A interface do SIM/SAVAD é a mesma definida para o SAVAD. Esta foi desenvolvida pelo Prof. Francisco de Assis Coutinho, como parte da sua dissertação de mestrado junto a Coordenação de Pós-Graduação em Informática (COPIN) da UFPB. Informações adicionais sobre o SAVAD podem ser encontradas em [Cabral 93b] e [Brasileiro 89].

1.2 - Contribuição Científica

Conforme [Cabral 93a], as tendências atuais para a construção de software de simulação baseiam-se em técnicas de inteligência artificial aplicadas na construção de **ambientes de simulação inteligentes**. Nestes ambientes, o usuário declara o conhecimento do sistema (especialmente, a declaração de objetos), define a meta desejada, e o ambiente trabalha para encontrar a solução. Na declaração do conhecimento sobre o sistema, o usuário define as classes de todos os modelos possíveis e/ou aceitáveis. Assim, é de responsabilidade do ambiente de simulação especialista achar, automaticamente, o modelo que preencha as especificações desejadas e procurar a solução adequada.

Um ambiente de simulação inteligente deve permitir aos usuários, facilmente, desenvolver e modificar representações dos sistemas reais ou hipotéticos; construir modelos, sem ter de gastar muitas horas aprendendo uma nova linguagem de modelagem; deve ter uma interface de entrada de dados que permita aos usuários fornecerem as informações descritivas do sistema ou entidades em estudo, sem exigir que eles conheçam a forma do modelo ou o formato eventualmente solicitado para os dados. Ao usuário deve ser permitido fazer modificações simples no modelo, sem que sejam necessárias modificações no projeto do sistema. Com este propósito, é que a base de conhecimento deve ser construída. Entre outros requisitos apresentados em [Shannon 86], o usuário deve poder selecionar os tipos e as formas de saída desejadas.

Um exemplo de ambiente de simulação inteligente é o SAVAD, já mencionado neste Capítulo.

O SIM/SAVAD é parte integrante do módulo solução do SAVAD. As facilidades oferecidas pelo SIM/SAVAD tornam-no uma ferramenta de simulação flexível podendo simular uma grande variedade de modelos de redes de filas. Neste sentido, dois elementos oferecidos pelo SIM/SAVAD são contribuições significativas no processo de modelagem de sistemas de redes de filas, particularmente, na modelagem de redes de computadores. Esse elementos, **ponto de sincronização** e **ponto escalonador** são, respectivamente, adequados à modelagem do mecanismo de controle de fluxo de redes de computadores e à modelagem de protocolos de acesso ao meio de redes locais de computadores.

Finalmente, o SIM/SAVAD vem contribuir no desenvolvimento científico do projeto orientado a objetos e da programação orientada a objetos, especialmente aplicados ao projeto e à implementação de programas simuladores.

1.3 - Organização da Dissertação

O restante desta dissertação está organizado da seguinte forma:

No Capítulo 2 são apresentados os elementos do SIM/SAVAD, que permitem a modelagem de sistemas de redes de filas, e a sua interface. Ainda, neste Capítulo, são apresentadas as medidas de desempenho que podem ser solicitadas ao SIM/SAVAD.

No Capítulo 3 é apresentada a descrição do SIM/SAVAD baseada na lógica de simulação acionada por eventos. Neste Capítulo são definidos os estados, os eventos, o relógio e as possíveis situações de bloqueio dos elementos do SIM/SAVAD.

No Capítulo 4 são apresentados o projeto orientado a objetos do SIM/SAVAD e considerações sobre sua implementação.

No Capítulo 5 é apresentado o processo de validação do SIM/SAVAD. São comparados resultados de modelos de filas simulados pelo SIM/SAVAD com aqueles obtidos através da utilização de técnicas analíticas, baseadas em Teoria das Filas, e também por resultados obtidos através da execução de alguns desses modelos em outros simuladores.

Finalmente, no Capítulo 6 são apresentadas as conclusões e sugestões de continuidade desta dissertação.

CAPITULO 2

MODELANDO
COM O
SIM/SAVAD

C A P I T U L O 2

MODELANDO COM O SIM/SAVAD

Neste Capitulo, inicialmente, são apresentados os elementos do SIM/SAVAD que permitem modelar sistemas de redes de filas. Em seguida, é apresentada a interface do SIM/SAVAD, que permite ao usuário:

a) construir e validar os modelos a serem simulados;

b) escolher niveis de confiança para as medidas de desempenho.

c) receber os resultados da simulação.

Finalmente, neste capitulo são apresentados as medidas de desempenho fornecidas pelo SIM/SAVAD.

2.1 Elementos de Modelagem

O SIM/SAVAD simula modelos de sistemas discretos que envolvem contenção de recursos. Para modelar tais sistemas, as entidades que solicitam recursos (serviços) são representadas por elementos denominados **clientes** e as entidades que prestam serviço são representadas por elementos denominados **nodos**.

Além dos elementos clientes e nodos, na descrição de um modelo, faz-se necessário definir quais os tipos de clientes que existem no modelo e quais os caminhos seguidos pelos clientes. Cada tipo é representado por um elemento denominado **classe** e os caminhos (sequências de nodos) são representados por elementos denominados **rotas**.

A seguir, é apresentada uma descrição dos elementos de modelagem do SIM/SAVAD:

- Clientes

Clientes são as entidades temporárias que circulam através dos **nodos** (elementos) do modelo solicitando serviços. Exemplos de entidades que podem ser representadas por clientes são os programas submetidos a um computador e as mensagens que são transmitidas em uma rede de computadores.

Detalhes sobre como o SIM/SAVAD faz os clientes circularem

através dos nodos são apresentados no Capítulo 3.

- Estações de Serviço

Estações de serviço são tipos de nodos que representam os recursos de um modelo de redes de filas. Estações de serviço podem ser constituídas de um ou mais servidores, que são entidades permanentes que atendem os clientes, um de cada vez, segundo uma **função de distribuição de serviço** (exponencial, uniforme, geral ou determinística).

Se o cliente recém-chegado encontrar a estação ocupada, ele entra na fila de espera. A um certo tempo será selecionado para serviço, de acordo com a **disciplina de escalonamento** "First Come, First Served" (FCFS).

As filas de espera podem ter comprimentos limitados ou ilimitados. Se uma fila de uma estação de serviço apresentar limitação de comprimento, este deverá ser indicado através do parâmetro **k** associado a esta fila.

Denominam-se **servidor simples**, **servidor múltiplo** e **servidor infinito**, respectivamente, aquelas estações de serviço que apresentam um servidor, múltiplos servidores e infinitos servidores.

Numa estação de serviço com infinito servidores cada cliente que chega tem sempre um servidor disponível para servi-lo.

- Pontos de Controle

Denominam-se pontos de controle aos nodos que permitem controlar o fluxo de clientes nos modelos de redes de filas. O SIM/SAVAD oferece as seguintes opções de pontos de controle:

1) **Ponto de Duplicação:** Possibilita a duplicação de clientes que chegam a este nodo. Possui uma entrada e duas saídas que podem pertencer a rotas distintas.

2) **Ponto de Fusão:** Faz a fusão de dois ou mais clientes em um único cliente. Associado a esse nodo existe um parâmetro **n** que determina o número de clientes que devem ser fundidos.

3) **Ponto de Sincronização:** Pode deter clientes, num modelo de redes de filas, até que uma dada condição seja satisfeita, quando então libera esses clientes com uma disciplina de liberaçao determinada pelo usuário. Um ponto de sincronização pode possuir:

a) duas entradas com duas saídas correspondentes;
b) duas entradas com somente uma saida correspondente a uma das entradas, e,
c) uma entrada com uma saída correspondente.

Nas alternativas (a) e (b), a condição para a liberação de clientes depende das filas de entrada do ponto de sincronização

alcançarem ambas os limites impostos pelo usuário. Esses limites são definidos através dos parâmetros **n1** e **n2**, respectivamente associados às entradas **1** e **2**. A disciplina de liberação de clientes é também definida por esses parâmetros, cujos possíveis valores são mostrados a seguir:

n1 (n2) : inteiro positivo (libera n1 (n2) clientes, um de cada vez).

-n1 (-n2): inteiro negativo (libera até n1 (n2) clientes, um de cada vez). Neste caso o parâmetro é um fator limitante (superior) para haver sincronização.

Um ponto de sincronização com apenas uma entrada e sua saída correspondente (alternativa c), possui apenas um parâmetro **n** a ele associado. Nessa alternativa, a condição de liberação de clientes resume-se ao número de clientes na fila de entrada alcançar o valor definido pelo parâmetro n (atribui-se para n os mesmos possíveis valores apresentados para n1 (n2)).

O número de rotas associadas a um ponto de sincronização é igual ao número de entradas deste tipo de nodo. No caso de um ponto de sincronização com duas entrada e apenas uma saída as rotas associadas são definidas da seguinte forma:

- uma das rotas possui uma sequência de nodos que termina no ponto de sincronização. Para esta rota, o ponto de sincronização elimina clientes após fazer sua sincronização com os clientes da outra rota.

- a outra rota possui uma sequência de nodos na qual o ponto de sincronização é um nodo intermediário.

4) **Ponto Escalonador**: escalona clientes conforme uma disciplina estabelecida pelo usuário. Um ponto escalonador pode possuir uma ou mais filas de clientes, conforme determinado pelo parâmetro **n** associado a este elemento.

As filas do ponto escalonador podem ter comprimento limitado ou não. No primeiro caso, este comprimento deve ser indicado através do parâmetro **k** associado ao conjunto de filas do ponto escalonador.

O ponto escalonador torna-se ativo quando o elemento do modelo que o segue não está bloqueado, isto é, este pode receber um cliente do ponto escalonador. Um ponto escalonador ativo escalona uma de suas filas podendo liberar um cliente desta fila de cada vez. Os clientes de uma fila são liberados conforme disciplina **FCFS**. Se uma fila escalonada não contiver um cliente, o ponto escalonador passa então para uma outra fila. O escalonamento de filas ocorre conforme uma das seguintes disciplinas:

a) **Cíclica**: filas são escalonadas conforme um vetor contendo uma sequência dos seus números associados.

Nessa sequência o número de uma fila pode constar mais de uma vez, e, após o escalonamento da fila correspondente ao último número, o escalonamento volta a ser feito a partir da fila associada ao primeiro número desta sequência.

O usuário pode variar a forma de liberação de clientes que deve ser utilizada em cada uma das filas da sequência. Estão a sua disposição as seguintes formas de liberação:

- não exaustiva: quando a fila é escalonada ela libera somente um cliente,
- limitada: quando a fila é escalonada ela libera até "n" clientes,
- exaustiva: quando a fila é escalonada todos os clientes da fila são liberados.

As duas últimas formas de liberação são especificadas, respectivamente, colocando-se entre parênteses o valor de "n" ou um asterisco (*) após o número que identifica a fila.

A seguinte sequência ilustra a forma de especificação de um escalonamento cíclico:

1(*), 2(2), 3, 2

Nessa sequência a fila 1 quando escalonada libera clientes de forma exaustiva; a fila 2 pode liberar até dois clientes; a fila 3 pode liberar um cliente, e, novamente a fila 2 pode voltar a liberar um cliente.

b) **Randômica**: nesta disciplina uma fila é escalonada de forma randômica, somente podendo liberar um cliente de cada vez. Um exemplo para este tipo de disciplina é a modelagem de uma aplicação envolvendo um multiplexador estatístico, no tempo.

c) **Livre**: nesta discipina, quando o ponto escalonador é ativado, qualquer fila que não estiver vazia pode tentar liberar um cliente. Se ocorrer colisão de clientes (dois ou mais clientes de filas distintas tentam avançar nas suas rotas em um mesmo tempo), estes ficam bloqueados nas respectivas filas por intervalo de tempo determinado por um algorítmo de retransmissão

- Fontes

Fontes (junto com os pontos de duplicação) são responsáveis pela geração de tráfego nos modelos de redes abertas de filas. Fontes geram clientes com tempos de interchegadas definidos por uma função de distribuição de probabilidade.

- Sorvedouros

Sorvedouros são nodos que eliminam clientes em modelos de redes abertas onde clientes eventualmente abandonam o modelo. Este nodo não possui parâmetros, possui apenas uma entrada.

- Rotas

Rotas estabelecem caminhos por onde clientes circulam. No SIM/SAVAD uma rota é especificada através da definição da sequência dos nodos constituintes, na ordem desejada e de forma fixa.

Rotas podem ser abertas ou fechadas. Rotas abertas necessitam de nodos que gerem e eventualmente eliminem clientes.

Rotas fechadas possuem um número fixo de clientes (população) que circula pela sequência de nodos. Neste caso, deve-se especificar a sequência de nodos e a população desta rota.

Deve-se observar que nodos associados a uma rota podem estar associados a outras rotas.

- Classes

Na definição de um modelo de redes de filas há a necessidade de definir as classes de clientes que circulam pelo modelo. Uma classe de clientes tem um nome e está associada a uma ou mais rotas possibilitando um tratamento diferenciado para um conjunto de rotas e dos clientes associados.

2.2 - Interface do SIM/SAVAD

A interface do SIM/SAVAD é amigável e constituida por janelas, menu principal e sub-menus verticais ("top-down") em que o usuário, de forma interativa, descreve seu modelo e recebe os resultados do processamento da simulação. Também fornece facilidades adicionais tais como manuseio de arquivos e acesso direto ao MS-DOS.

Durante a descrição de um modelo, a interface automaticamente valida cada um dos elementos que compõem o mesmo, permitindo ao usuário acessar um sistema de ajuda ("help") sensível ao contexto, caso necessário.

Segue uma descrição sucinta da interface do SIM/SAVAD. Mais informações podem ser encontradas em [Cabral 92]

As seguintes opções são fornecidas no menu principal:

- Arquivo

Permite a execução de operações de manipulação dos arquivos usados para a descrição do modelo.

Esta opção possibilita o acesso a um arquivo já existente, criar um novo arquivo, gravar, eliminar, copiar ou renomear arquivos, além de exibir em tela o conteúdo de diretórios.

- Edição

Corresponde ao processo de definição do modelo através da especificação dos seus diversos elementos.

Esta opção permite a inserção, alteração, retirada e cópia de elementos do modelo bem como a modificação dos nomes destes elementos.

- Consulta

Permite a consulta da configuração do modelo, do resultado de sua verificação de integridade ou do resultado da simulação.

A consulta pode ser feita individualmente a cada elemento, a uma categoria de elementos, ao modelo completo ou apenas ao resumo do modelo (número de elementos por categoria e informações sobre a verificação da integridade do modelo).

A consulta pode ter o seu resultado apresentado na tela ou impresso.

- Verificação

Faz a verificação da integridade da especificação do modelo. A verificação é feita antes da simulação para evitar que esta seja efetuada com informações inconsistentes. Esta opção verifica os seguintes ítens:

a) Elementos referenciados em rotas mas não definidos.
b) Elementos não encadeados adequadamente na rota.
c) Elementos definidos mas não usados nas rotas do modelo.
d) Rotas abertas sem fonte (ponto de duplicação)/sorvedouro (ponto de sincronização e ponto de fusão).
e) Rotas fechadas com o somatório dos comprimentos de filas inferior a sua população de clientes.

- Processa

Faz a verificação do modelo, caso não tenha sido solicitada pelo usuário e, se este foi especificado corretamente, ativa o processamento da simulação. Esta opção solicita ainda informações sobre os parâmetros necessários à simulação, tais como condições de término da simulação e informações sobre níveis de confiança para as medidas de desempenho e sobre o número desejado de simulações.

- Opções/Comandos DOS

Permite ao usuário executar diretamente comandos do Sistema Operacional.

- Sair

Neste caso o processamento do SIM/SAVAD é encerrado e o

programa retorna ao Sistema Operacional, salvando eventuais alterações feitas no modelo.

As figuras de 2.1 a 2.6 mostram exemplos de telas da interface. Excetuando as tela da figura 2.1 e 2.2 em todas as outras telas o usuário pode solicitar ajuda sensível ao contexto, para cada ítem, através do acionamento da tecla **F1**. Acionando a tecla **F10** o trabalho ora efetuado é cancelado e ocorre um retorno ao menu principal.

A figura 2.1 mostra a tela com o submenu aberto com a escolha da opção **Edição**. Caso nesta tela seja escolhida a opção **Inserir elemento** a interface apresenta a tela mostrada na figura 2.2, isto é, um submenu para selecionar um elemento de modelagem.

Caso na tela mostrada pela figura 2.2 seja escolhido um elemento do tipo Servidor, o SIM/SAVAD apresenta a tela mostrada na figura 2.3, isto é, uma tela para informar os parâmetros de relacionados a uma estação de serviço.

A figura 2.4 mostra o que acontece se na tela mostrada pela figura 2.2 for escolhida a inclusão de um elemento do tipo Rota. Neste caso o SIM/SAVAD apresenta a tela mostrada na figura 2.3, isto é, uma tela para informar os parâmetros relacionados a uma rota. Observe que ao especificar "pontoescalonador.1" está se definindo que os clientes dessa rota devem se dirigir para a fila 1 do nodo denominado pontoescalonador.

A figura 2.5 mostra a tela ativada pela opção **Processa** no menu principal. Nesta tela o usuário informa os parâmetros que serão utilizados na simulação para calcular as medidas de desempenho.

Para ativar o simulador o usuário deve informar que:

a) o tipo de processamento é por simulação,
b) a condição de término é por tempo e qual o total de unidades de tempo simulado que limita a simulação,
c) o nível de confiança desejado, o qual será utilizado no método usado para cálculo do intervalo de confiança [Moura 86].
d) o número de simulações a serem executados e a semente que deve ser utilizada em cada uma dessas simulações.

Para cada elemento de modelagem que utilizar uma distribuição que não seja do tipo determinística, o SIM/SAVAD cria um gerador de amostras para a distribuição definida. A semente associada no item (d) é usada na geração de números aleatórios.

A figura 2.6 mostra a tela ativada pela opção **Consulta** no menu principal. Nesta tela, se o usuário selecionar a opção **Solução**, o SIM/SAVAD mostra os resultados coletados após ativar a opção **Processa**.

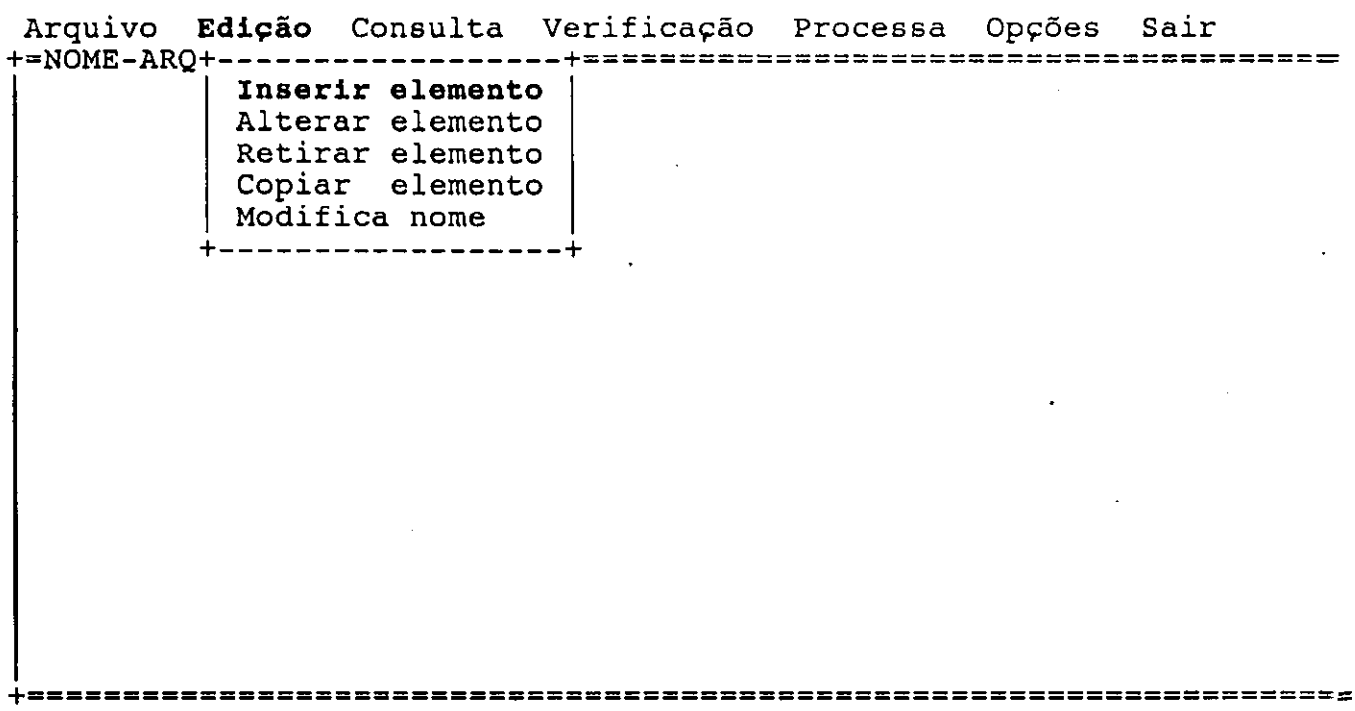


Figura 2.1 - Submenu Aberto Pela Opção Edição

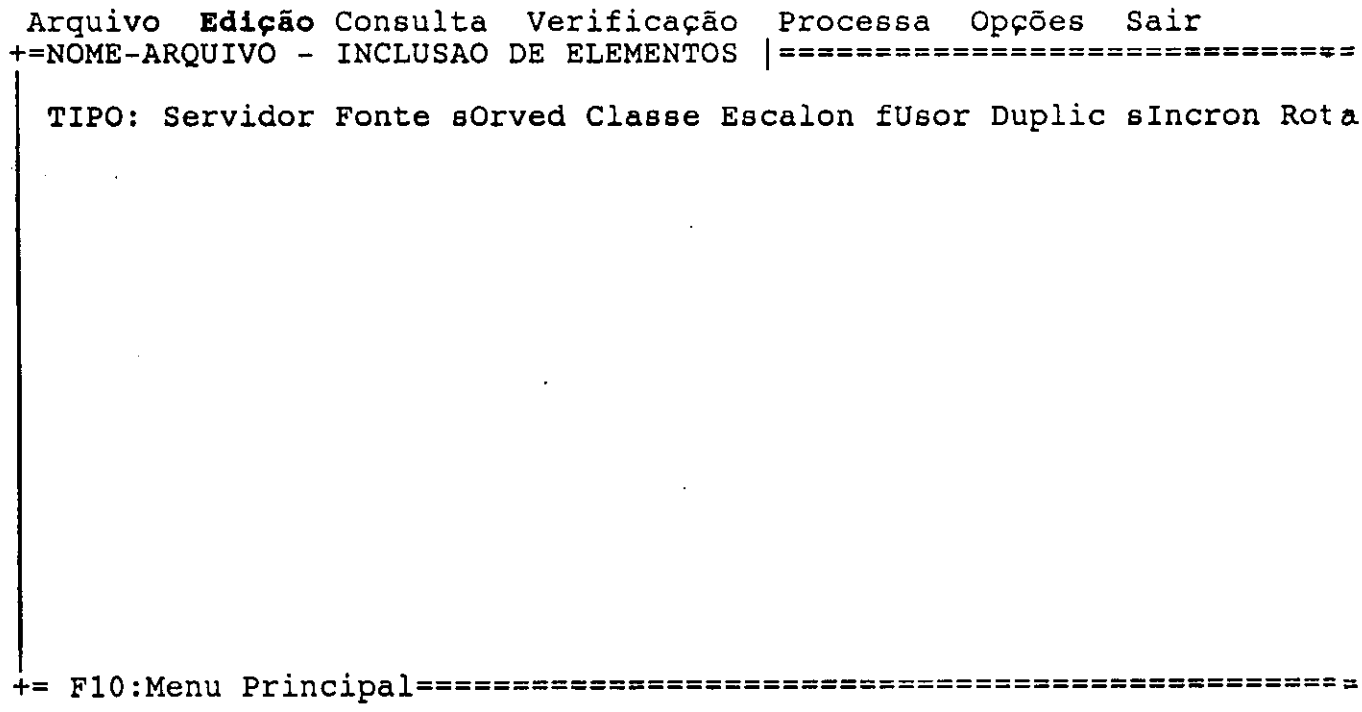


Figura 2.2 - Submenu Para Escolha do Tipo de Elemento a Ser Inserido

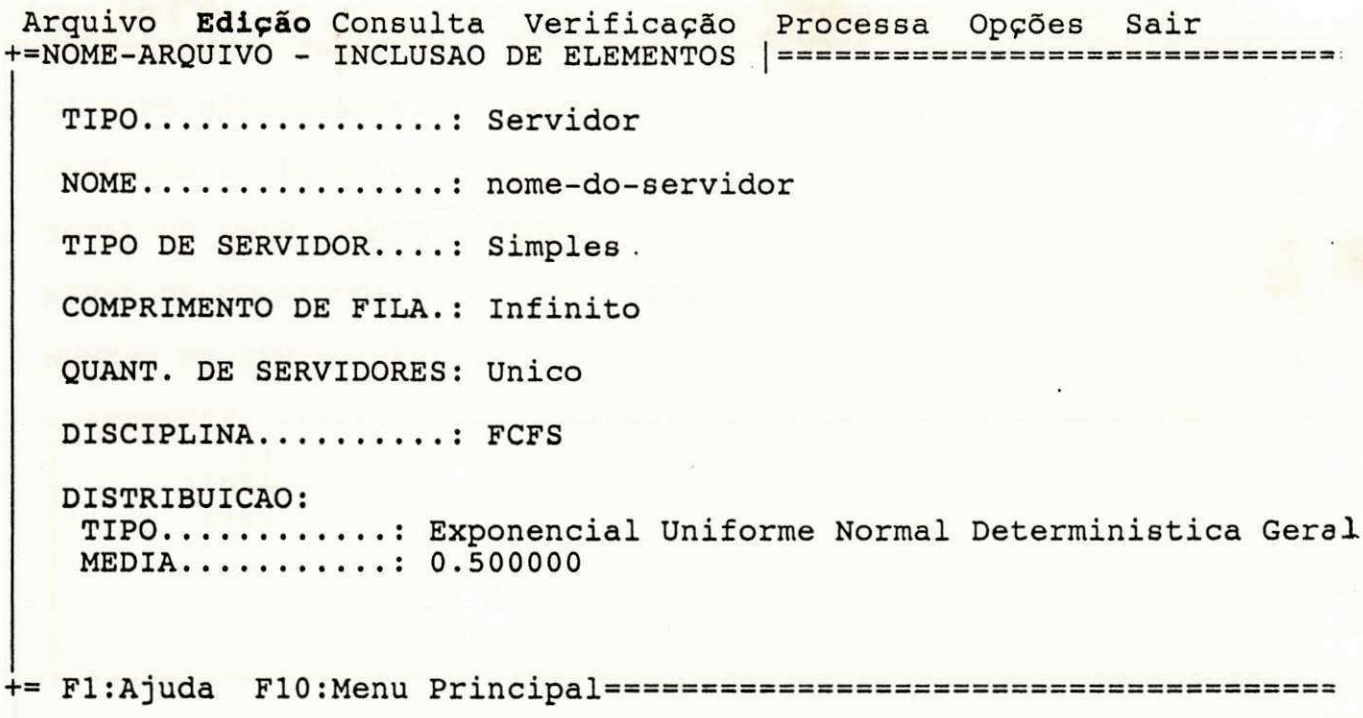


Figura 2.3 - Tela Para Especificação de Estação de Serviço

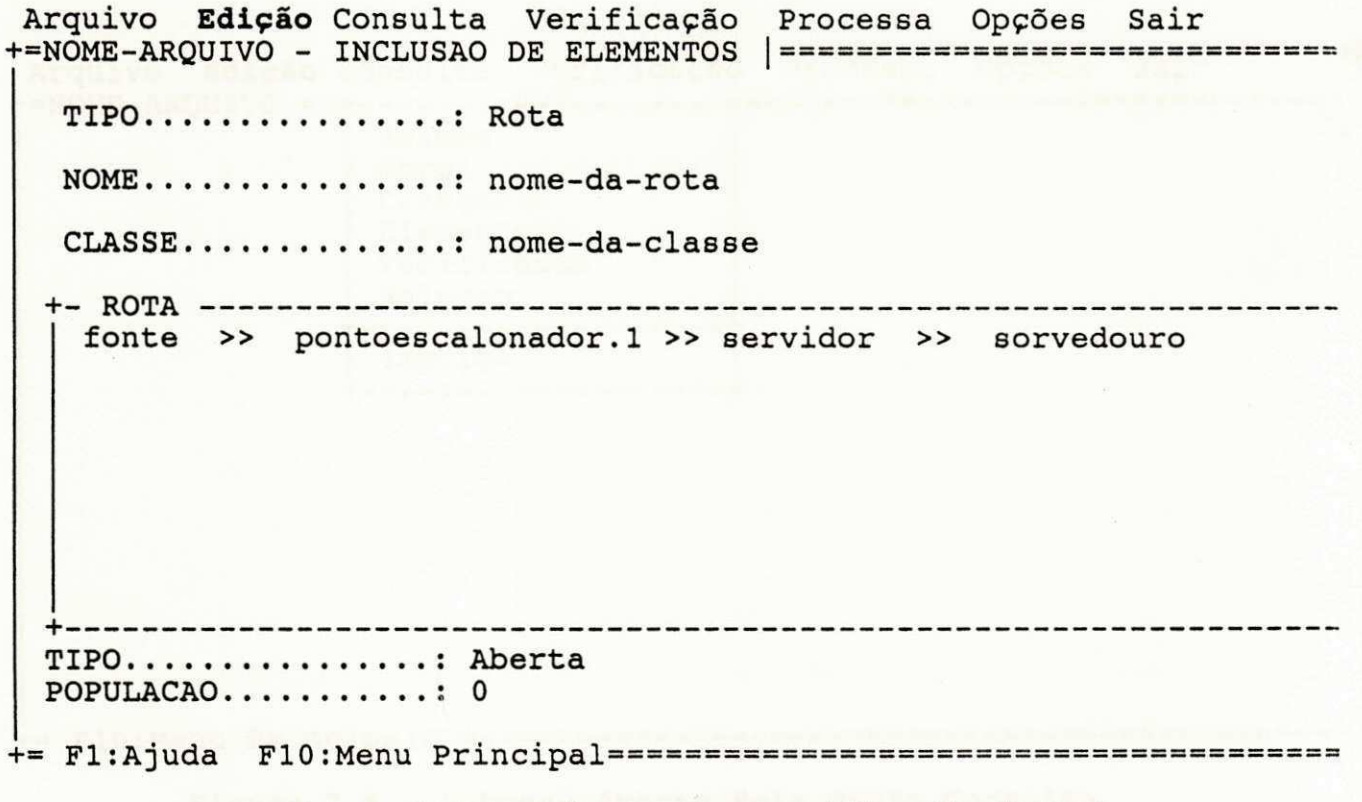


Figura 2.4 - Tela Para Especificação de Rota

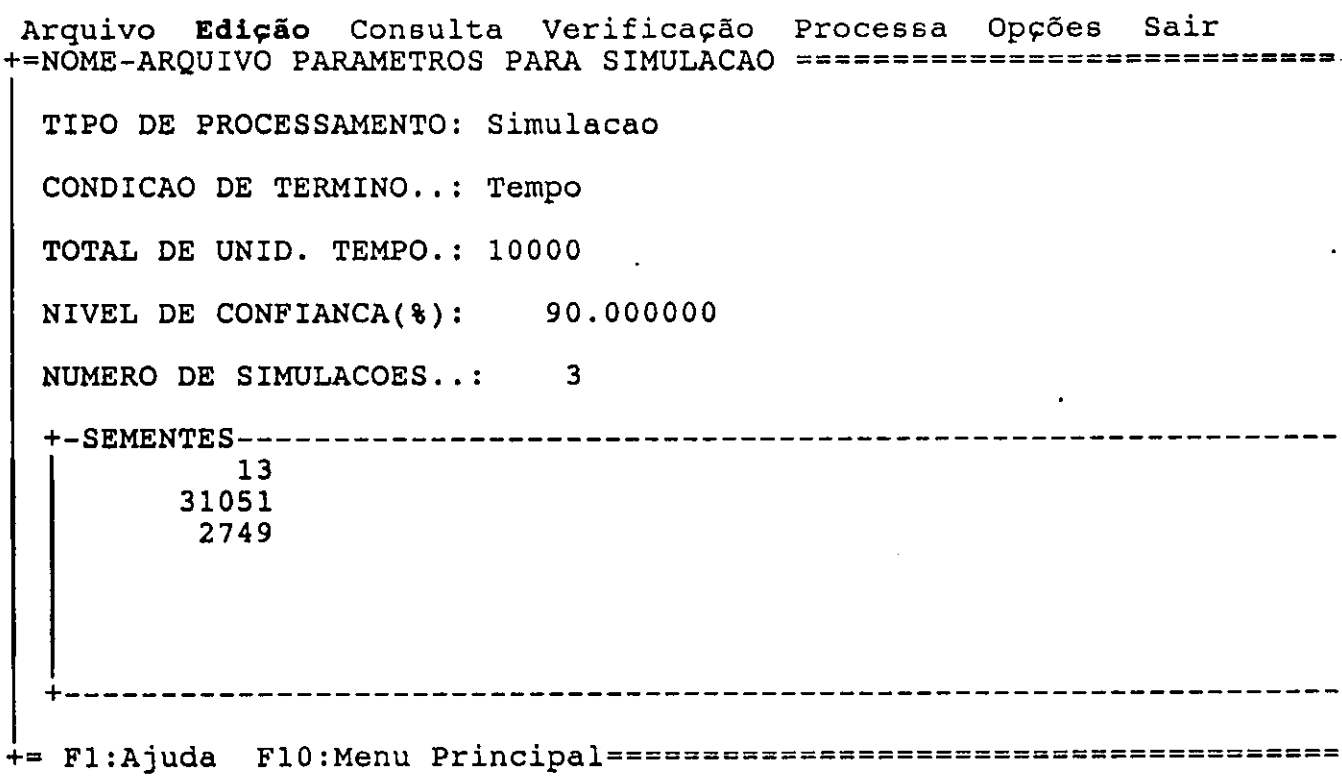


Figura 2.5 - Tela Para Especificação de Parâmetros Para Simulação

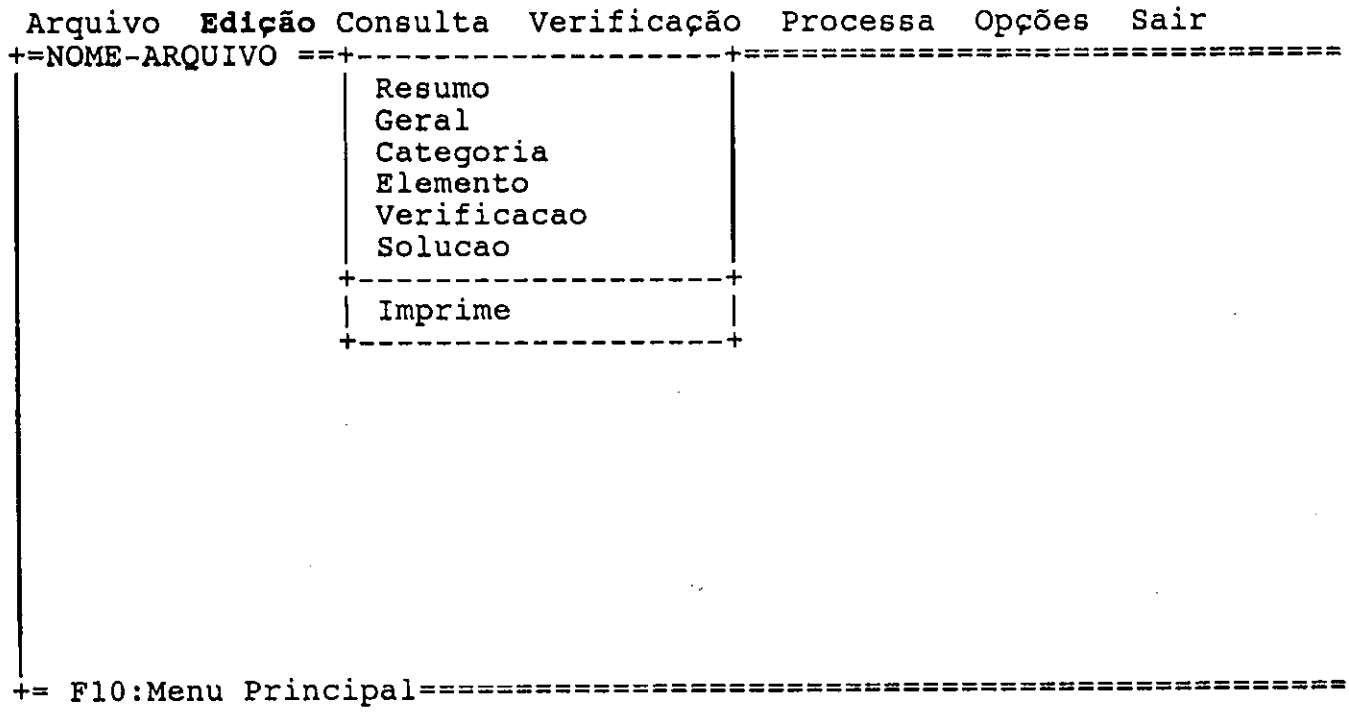


Figura 2.6 - Submenu Aberto Pela Opção Consulta

2.3 - Medidas de Desempenho

Como resultado de uma simulação, o SIM/SAVAD gera um arquivo com uma série de medidas de desempenho. Este arquivo pode ser exibido com a opção **Consulta** da interface.

Seguem as medidas de desempenho oferecidas pelo SIM/SAVAD conforme estejam associadas a uma das seguintes categorias:

a) nodos distintos,
b) cadeias de filas e,
c) ao modelo de rede de filas.

a) Nodos:

- Fonte:

  - número de clientes gerados.
  - número de interrupções de geração de clientes (bloqueios).

- Sorvedouros

  - número de clientes eliminados por rota.
  - número de clientes eliminados no modelo.

- Ponto de Duplicação:

  - número de duplicações feitas.
  - número de duplicações adiadas devido a situação de bloqueio.
  - atraso médio de duplicações adiadas (bloqueadas).

- Ponto de Fusão:

  - número de fusões feitas.
  - número de fusões adiadas devido a situação de bloqueio.
  - atraso médio de fusões adiadas (bloqueadas).

- Nodos que possuem fila (estações de serviço, pontos de sincronização e pontos escalonadores):

  - Para cada fila:

    - comprimento máximo de fila.
    - comprimento corrente de fila.
    - comprimento médio de fila.
    - número médio de clientes que chegaram a fila.
    - tempo médio de fila.
    - número de clientes que foram servidos e não esperaram em fila.

  - Estação de Serviço:

    - número total de clientes que chegaram a estação.

- número total de clientes que foram servidos.
- número total de clientes que ficaram bloqueados.
- número corrente de clientes em serviço.
- número corrente de clientes bloqueados.
- número médio de clientes na estação (clientes em fila mais clientes em serviço).

- Ponto de Sincronização:

  - número de sincronizações feitas.
  - número de sincronizações adiadas devido a situações de bloqueio.
  - atraso médio de sincronizações adiadas (bloqueadas).

- Ponto Escalonador:

  - número de escalonamentos feitos.
  - número de escalonamentos adiados devido a situações de bloqueio.
  - atraso médio de escalonamentos adiados (bloqueados).

b) Cadeias de Filas

- Cadeia Aberta:

  - número de clientes gerados.
  - número de clientes descartados.
  - vazão média
  - atraso médio fim-a-fim.

- Cadeia Fechada:

  - vazão média.
  - atraso médio fim-a-fim.

c) Do Modelo:

  - número de clientes gerados.
  - número de clientes descartados.
  - vazão média
  - atraso médio fim-a-fim.

2.4 - Mecanismos para Execução de um Modelo

Na execução de um modelo, o SIM/SAVAD usa um conjunto de mecanismos que gerenciam os seguintes itens:

- **relógio simulado**: a simulação é dinâmica, sendo necessário mecanismos que representem a evolução do tempo, para isto o SIM/SAVAD utiliza o relógio simulado. No capítulo 3 este mecanismo é detalhado.

- **bloqueio**: se todos os clientes de um modelo ficarem bloqueados, o SIM/SAVAD interrompe a simulação. A gerência de bloqueios é detalhada no capítulo 3.

- **simulação de aleatoriedade:** os elementos de modelagem do SIM/SAVAD podem trabalhar com taxas fixas ou não. Para proporcionar taxas que variam aleatoriamente, o SIM/SAVAD dispõe de geradores de números aleatórios e de métodos numéricos que fornecem amostras das funções de distribuição de probabilidade exponencial, uniforme e geral.

- **geração de sementes:** para cada elemento de modelagem que necessita amostras de funções de probabilidade, o SIM/SAVAD define um gerador de números aleatórios. Para cada gerador o SIM/SAVAD atribui uma semente.

- **limites da simulação:** quando maior for o tempo de simulação de um modelo, maior deve ser a precisão das medidas que forem obtidas. Por outro lado, deve-se limitar a duração da simulação, para isto o usuário do SIM/SAVAD define um limite de tempo simulado através da opção **Processa** da interface.

- **intervalos de confiança:** o usuário do SIM/SAVAD pode solicitar que seja executada mais de uma simulação para um modelo.

Nesse caso, em cada uma das execuções são modificadas as sementes utilizadas na geração de números aleatórios. Os valores obtidos para as medidas de desempenho em uma execução são tratados como uma observação e um conjunto de observações é usado para encontrar os valores médios das medidas de desempenho para um nível de confiança desejado.

O usuário pode escolher resultados com os seguintes níveis de confiança 90% ou 99%. Esta escolha é feita na opção **Processa** da interface.

As medidas de desempenho obtidas nas "n" execuções são apresentadas por seu valores médios. Também são fornecidas, para cada medida, além das médias amostrais destas, o desvio padrão, o valor obtido da tabela da distribuição de Student-t a partir do nível de confiança escolhido e o intervalo de confiança determinado segundo o método descrito em [Moura 86].

As medidas de desempenho que são fornecidas com nível de confiança são as seguintes:

- tempo médio de resposta de uma estação de serviço (tempo médio de espera em fila mais tempo médio de serviço).

- utilização da estação de serviço (para as estações do tipo servidor simples e do tipo servidor múltiplo).

- número médio de clientes em estação de serviço do tipo servidor infinito.

- tempo médio de permanência dos clientes no modelo.

- tempo médio de espera nas filas de um ponto escalonador.

- tempo médio de espera em cada uma das filas de um ponto de sincronização.

CAPITULO 3

SIMULACAO
ACIONADA
POR EVENTOS

C A P Í T U L O 3

SIMULAÇÃO ACIONADA POR EVENTOS

Neste Capítulo é descrito o SIM/SAVAD segundo a lógica de simuladores acionados por eventos.

As Seções que seguem apresentam a terminologia usada neste Capítulo; as descrições dos Estados, do Relógio Simulado e dos Eventos do SIM/SAVAD. Finalmente situações de bloqueios de clientes e de nodos são descritas.

3.1 Simulação Acionada por Eventos

Os diversos módulos de um simulador acionado por eventos são executados em resposta (direta ou indireta) à ocorrência de "eventos". Um evento é uma "perturbacao" instantânea que muda o estado do sistema (e por conseguinte, do modelo).

Conforme [Moura 86:292] deve-se:

- "identificar os eventos que podem ocorrer no sistema;

- avaliar os efeitos dos eventos no estado e atividades do sistema;

- permitir a ocorrência dos eventos e atualizar o estado e atividades do sistema à medida que os eventos ocorrem."

3.2 - Terminologia

Neste Capítulo e nos seguintes alguns termos são frequentemente usados. Para propiciar ao leitor um melhor entendimento, seguem descrições desses termos.

- **nodo corrente**: é o nodo que estiver associado ao evento escalonado para processamento pelo SIM/SAVAD.

- **entrada de um nodo**: fila ou local associado a entrada de um nodo.

- **nodo seguinte**: nodo que segue o nodo corrente numa rota.

- **nodo anterior**: nodo que precede o nodo corrente numa rota.

- **avanço de cliente**: significa que o cliente passa do nodo

corrente para o nodo seguinte.

- **cliente bloqueado:** significa que este cliente não pode avançar na sua rota porque o nodo seguinte está impossibilitado de receber novos clientes.

3.3 Estados

Os estados mais importantes do SIM/SAVAD são os seguintes:

- **nodo não disponível:** se for um nodo que possui fila significa que a capacidade desta está esgotada. Se for um ponto de duplicação significa que pelo menos um de seus nodos seguintes está no estado não disponível. Se for um ponto de fusão, significa que o nodo seguinte está no estado não disponível ou que não houve chegada de clientes em suas entradas que permitissem sua ativação.

- **nodo disponível:** significa que o nodo pode receber novos clientes.

- **estação de serviço ocupada:** significa que todos os seus servidores estão ocupados atendendo clientes.

- **estação de serviço livre:** significa que esta possui pelo menos um servidor desocupado.

- **ponto escalonador inativo:** significa que não há clientes na(s) sua(s) fila(s).

- **ponto de sincronização inativo:** significa que a quantidade de clientes na(s) fila(s) deste nodo não é suficiente para permitir uma sincronização de clientes.

- **fonte ativa:** significa que a fonte está gerando clientes segundo a Função de Distribuição de Probabilidade definida para seu processo de geração de clientes.

3.4 - Relógio Simulado

"Num simulador acionado por eventos, o relógio deve ser sempre avançado para o instante de ocorrência do próximo evento, isto é, depois que o simulador atualizou o estado do modelo em resposta à ocorrência de um evento, é necessário cuidar do evento que é o próximo a ocorrer (evento iminente). Para tanto, a primeira medida a ser tomada é adiantar o relógio para o instante de ocorrência do evento iminente" [Moura 86:296].

No SIM/SAVAD a forma utilizada para adiantar o relógio simulado é apresentada na figura 3.1

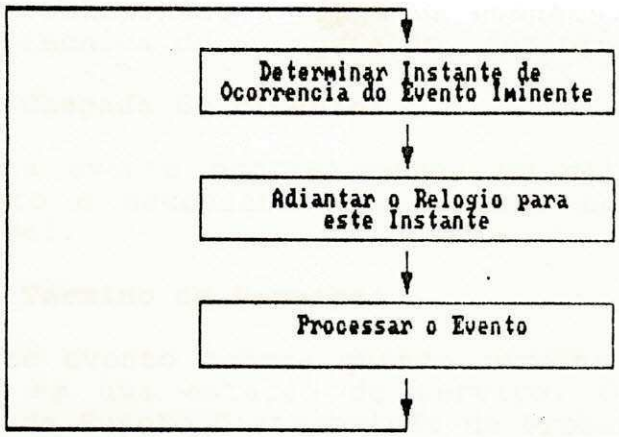


Figura 3.1 - Atualizacao do Relogio Simulado

3.5 - Eventos

Em simulação acionada por eventos, o estado do sistema só pode mudar nos tempos de ocorrência de **eventos**. Dessa forma, eventos são ocorrências que mudam o estado do sistema.

Um programa-simulador "... é um agrupamento de procedimentos ou rotinas que definem como o estado do modelo muda quando da ocorrência de cada evento. Para que os diversos eventos acionem as várias rotinas, e para que a execução do simulador seja continuada, os eventos devem ser escalonados devidamente." [Moura 86:293].

Para organizar os eventos mantendo sua ordem cronológica de ocorrência o SIM/SAVAD dispõe de uma **lista de eventos e uma lista de desbloqueio**; os eventos destas listas são escalonados conforme sua ordem de ocorrência.

O processo de escalonamento de eventos do SIM/SAVAD é conforme [Sauer 81], [Goldberg 83] e [Cauper 86]. Esse processo de escalonamento pode ser resumido nas seguintes etapas:

a) escalona o evento mais iminente da lista de eventos,

b) atualiza o relógio simulado com o tempo de ocorrência do evento escalonado, e

c) execução de ações: consiste na execução das ações associadas com a ocorrência do evento escalonado. As ações a serem executadas dependem do tipo do evento e das condições do modelo (estados) no momento da ocorrência desse evento.

Os tipos de eventos do SIM/SAVAD são os seguintes:

a) **Chegada de cliente originado em fonte:**

Fontes são nodos que podem gerar clientes continuamente, segundo tempos de geração obtidos de amostras de uma Função de Distribuição de Probabilidade especificada pelo usuário. Para cada cliente gerado um evento **chegada de cliente originado em**

fonte é adicionado a lista de eventos. A geração desses eventos segue a técnica de autogeração, conforme [Moura 86:293]

b) **Chegada de cliente:**

Este evento ocorre quando um cliente avança na sua rota. Para isto é necessário que o nodo seguinte esteja no estado disponível.

c) **Término de Serviço:**

Este evento ocorre quando termina o tempo de serviço de um cliente em uma estaçao de serviço. O tempo de serviço é uma amostra da Função Distribuição de Probabilidade especificada pelo usuário.

Após o processamento do evento **término de serviço**, o SIM/SAVAD verifica se o cliente associado a este evento pode avançar na sua rota, para isto ele verifica se o próximo nodo está no estado disponível. Em caso positivo, é gerado o evento **chegada de cliente em um nodo**. Em caso contrário, o cliente fica bloqueado (ficando impossibilitado de avançar na sua rota). Nesse caso, o servidor que atende o cliente também fica bloqueado.

Se houver um cliente na fila da estação de serviço, o SIM/SAVAD gera um novo evento **término de serviço**. Esta é mais uma aplicação da técnica de autogeração: "... quando ocorre um ´término de serviço´, um novo evento deste tipo nem sempre é escalonado; apenas quando há pelo menos um freguês esperando serviço é que outro ´término de serviço´ será escalonado." [Moura 86:293].

d) **Nodo Disponível:**

Este evento ocorre quando um nodo que está bloqueando clientes passa para o estado disponível.

e) **Fim de Simulação:**

Este evento representa o término da simulação, que deve ocorrer num tempo especificado pelo usuário.

3.6 - Situações de Bloqueio

Nodos que estejam no estado não disponível podem bloquear o fluxo normal de clientes. Doravante será denominado **bloqueador** o nodo que causar esta situação.

No momento que um cliente, posicionado em um dado nodo (nodo corrente), é liberado por este para avançar na sua rota, o SIM/SAVAD verifica se o nodo seguinte está no estado disponível. Em caso positivo gera o **evento chegada de cliente** associado ao nodo seguinte; em caso contrário o SIM/SAVAD bloqueia o cliente, podendo, conforme o tipo do nodo, bloquear o nodo corrente.

Doravante será denominado **bloqueado** o nodo que possuir cliente(s) bloqueado(s). Devido a possibilidade de um nodo ser compartilhado por diversas rotas, um nodo bloqueado pode ter clientes bloqueados por mais de um nodo seguinte (nodo bloqueador).

Para desbloquear nodos, o SIM/SAVAD verifica continuamente se algum nodo bloqueador passa do estado não disponível para o estado disponível, isto é, se ocorre algum evento nodo disponível.

Para viabilizar esse processo o SIM/SAVAD dispõe de **listas de bloqueio** nas entradas dos nodos bloqueadores. Nessas listas são armazenadas as relações de clientes e de nodos bloqueados. Somente nodos que são ponto de controle são colocados nas listas de bloqueio.

No caso de um nodo bloqueado receber um evento nodo disponível e possuir cliente(s) bloqueado(s), o SIM/SAVAD verifica se algum cliente pode avançar e em caso positivo gera o **evento chegada de cliente** associado ao nodo que gerou o **evento nodo disponível.**

O SIM/SAVAD deve verificar se, após o desbloqueio do(s) cliente(s), outros clientes podem avançar para o nodo seguinte (antes nodo bloqueador) neste mesmo momento; em caso positivo a transferência de clientes do nodo anteriormente bloqueado para o nodo seguinte é feita.

Caso contrário, ou não havia mais clientes bloqueados, ou o nodo seguinte voltou a se tornar bloqueador para os clientes restantes.

Deve-se observar que pontos de duplicação e pontos de fusão não podem armazenar clientes bloqueados. Se o nodo seguinte é um nodo bloqueador, os nodos anteriores armazenarão os clientes bloqueados.

Um nodo deixará de ser nodo bloqueado quando o **evento nodo disponível** associado ao nodo seguinte ocorrer.

Finalizando esta Seção, descrevem-se as ações que o SIM/SAVAD executa quando um cliente é bloqueado. Essas ações são apresentadas a seguir, conforme o tipo do nodo corrente associado:

1) Fonte:

Neste caso há um reescalonamento da chegada do cliente conforme a Função Distribuição de Probabilidade especificada pelo usuário.

2) Estação de Serviço (servidor simples e servidor múltiplo):

O cliente que foi bloqueado continua ocupando o servidor que o atendeu (mantendo-o bloqueado). Este cliente espera que o nodo seguinte passe para o estado disponível.

3) Ponto Escalonador:

Se o nodo seguinte passar para o estado não disponível e ainda existir cliente numa fila do ponto escalonador, este passa a ser um nodo bloqueado.

4) Ponto de Sincronização:

Se quando ocorre uma sincronização, o(s) nodo(s) seguinte(s) encontra(m)-se no estado não disponível, os clientes do ponto de sincronização que poderiam avançar devem esperar o(s) evento(s) do tipo nodo disponível.

CAPITULO 4

PROJETO E IMPLEMENTACAO DO SIM/SAVAD SEGUNDO O PARADIGMA DE OBJETOS

C A P Í T U L O 4

Projeto e Implementação do SIM/SAVAD segundo o Paradigma de Objetos

No desenvolvimento do SIM/SAVAD foi explorada a potencialidade do paradigma de objetos. Este paradigma fornece uma nova metodologia para projetar e implementar software objetivando incrementar a produtividade do programador fornecendo um "framework" que enfatiza a extensibilidade e a reutilização de software [Diesch 91].

Os paradigmas de programação são modelos de como projetar e implementar programas [Dewhurst 90:2]. Novos paradigmas surgem devido às mudanças no modo como as pessoas vêm a atividade de programar. A base do paradigma de objetos parte da assertiva de que o mundo real é composto por objetos e não por dados ou processos, devendo o desenvolvimento de software estar baseado na identificação e manipulação de objetos.

Neste Capítulo é apresentado a utilização do paradigma de objetos no desenvolvimento do SIM/SAVAD. Nas seções que seguem são introduzidos os conceitos mais relevantes deste paradigma e o projeto orientado a objetos do SIM/SAVAD, apresentando as hierarquias de classes e o modelo cliente-servidor. Finalizando, apresentam-se comentários sobre o uso de polimorfismo e amarração dinâmica no desenvolvimento do SIM/SAVAD e sobre o uso da linguagem C++ na sua implementação.

4.1 - Paradigma de Objetos

Desde os princípios dos anos 80, o chamado **paradigma de objetos** tem atraído grande atenção de pesquisadores e projetistas de software como sendo a grande evolução desde as idéias de programação estruturada dos anos 70 [Rentsch 82].

Independente da estrutura que tome a implementação de um sistema, a primeira grande tarefa em desenvolvimento de software é a análise do **domínio de aplicação** e a modelagem das entidades e fenômenos desse domínio que o projetista considerar relevantes para a aplicação [Takahashi 90:13].

Na observação de fenômenos do domínio de aplicação deve-se considerar três conceitos:

- **categoria**: a partir de um fenômeno pode-se descrever uma categoria de fenômenos. Desta forma o fenômeno observado é dito

uma instância **concreta**, enquanto a categoria descreve um padrão (**arquétipo**) através de suas propriedades.

- **atributo:** é uma propriedade descritiva do fenômeno, isto é, uma informação associada a uma característica do mesmo.

- **ação:** é uma propriedade que o fenômeno possui de ser transformado. Um fenômeno pode mudar seu estado (seus atributos) ou gerar um novo fenômeno devido a uma ação.

Analisando os elementos de modelagem do SIM/SAVAD, pode-se exemplificar a observação de categoria de entidades no domínio da aplicação pela definição de uma categoria que abranja todos os elementos fontes. Cada instância da categoria fonte tem seu próprio identificador (atributo) e todas podem cumprir a função (ação) de gerar clientes.

No caso do paradigma de objetos a primeira e principal característica é a visão do domínio de uma aplicação como composto por **objetos** que se comunicam através de **mensagens** [Takahashi 90:64].

Desta forma as entidades e fenômenos do domínio são associados com objetos, e as ações observadas no domínio de aplicação são associadas às respostas dos objetos às mensagens recebidas. Um objeto que recebe uma mensagem responde a ela através da seleção e da execução de um **método** que fará parte de seu comportamento.

Objetos de estrutura e comportamento idênticos são descritos como **instâncias** de **classes**, de tal sorte que a descrição de suas propriedades pode ser feita de uma só vez, de forma concisa, independente do número de objetos idênticos em termos de estrutura e comportamento que possam existir em uma aplicação. Assim o conceito de descrever categorias está associado com a definição de classes que encapsulam as propriedades dos objetos (ações e atributos), e o conceito de descrever ações está associado a definição de mensagems e métodos.

Entre as facilidades oferecidas pelo paradigma de objetos estão o polimorfismo e os mecanismos de herança e amarração dinâmica.

- **Polimorfismo** ou sobrecarga de seletores consiste na possibilidade de múltiplos métodos compartilharem um mesmo identificador (seletor).

- **Herança** é a possibilidade de uma classe ser definida em função de outras classes, ou seja, a possibilidade de uma classe herdar características de classes anteriormente definidas. Neste caso a classe que herda propriedades se subordina a outra numa **hierarquia** de classes, passando a ser denominada como **subclasse** e a outra como **superclasse**.

Os níveis mais baixos de uma hierarquia de classe usualmente

Os níveis mais baixos de uma hierarquia de classe usualmente representam uma especificação mais detalhada, e níveis mais altos normalmente representam mais simplicidade [Pinson 91:9].

Outro ponto interessante é que na maioria das linguagens orientadas a objeto, se a classe P é superclasse da classe S, então um objeto s, da classe S, pode ser usado em qualquer lugar que um objeto p, da classe P, pode ser usado. Isto implica que um conjunto de mensagens comuns pode ser enviado a objetos da classe P e da classe S.

Numa hierarquia de classes um identificador de método pode estar associado a uma ação e num nível mais baixo da hierarquia pode estar associado a uma outra ação com características diferentes. O polimorfismo permite a cada objeto responder a um identificador de mensagem numa maneira apropriada à classe que o objeto pertence.

- **Amarração Dinâmica** está associada as duas facilidades anteriores. Devido à definição de hierarquias de classes pode-se ter acesso a um objeto com apontadores ou referências que não correspondem ao tipo real do objeto. Neste caso, a amarração dinâmica permite o acesso aos métodos da classe real do objeto quando apontadores para este acesso são de classes de nível superior na hierarquia de classes.

O desenvolvimento de software usando este novo paradigma ainda carece da padronização de metodologias. Segundo [Coimbra 91:24] "A maior dificuldade encontrada no desenvolvimento de sistemas orientados para objetos é a inexistência de uma metodologia específica. O desenvolvimento de sistemas orientados para objetos deve levar em consideração a necessidade de se definir uma metodologia de trabalho adequada à implementação dos conceitos de orientação a objetos."

Segundo [Takahashi 90:76] um programador conduzirá uma tarefa típica de programação da seguinte forma:

- o programador conceberá um modelo (ou parte de um modelo) para a sua aplicação, constituído por objetos se comunicando por mensagens.

- para cada conjunto de objetos com características idênticas, o programador abstrairá uma classe que o descreverá adequadamente.

- a descrição da classe será feita na **linguagem** disponível, onde uma parte das propriedades da classe recém-definida será **herdada** de uma superclasse previamente existente na **biblioteca de classes**.

- um programa será composto por uma classe cuja instanciação será iniciada pelo programador (caracterizando uma espécie de programa principal) e por todas as classes que serão ativadas direta ou indiretamente pela classe primeira.

Quando o programa estiver em execução, objetos serão instanciados a partir de classes, objetos enviarão mensagens a outros objetos, e as respostas de objetos a mensagens recebidas comporão a ação propriamente dita do programa.

4.2 - Projeto Orientado a Objetos do SIM/SAVAD

No Projeto do SIM/SAVAD foram definidas 45 (quarenta e cinco) classes. O apêndice A apresenta os protocolos dessas classes, isto é, a descrição de todas as classes do SIM/SAVAD, com seus métodos e variáveis (atributos).

As Subseções que seguem apresentam o projeto orientado a objetos do SIM/SAVAD através das seguintes representações: hierarquia de classes, modelos cliente-servidor e extensões desses modelos para as fases de processamento do SIM/SAVAD.

4.2.1 - Hierarquia de Classes

Nesta Subseção são apresentadas as 32 (trinta e duas) classes que foram organizadas em hierarquias. As outras classes, não pertencem a nenhuma hierarquia. Segundo [Takahashi 91:111] este tipo de classe é denomidada **classe básica**.

Inicialmente foi definido uma hierarquia de classes para representar nodos. Nesta hierarquia a superclasse foi denominada NodoVirtual e há uma subclasse para cada tipo de nodo. No caso das estações de serviço, foi definida uma classe para estações com um número limitado de servidores e outra para estações do tipo servidor infinito.

Devido ao mecanismo de herança, pode-se, por exemplo, usar um método que está definido na classe NodoVirtual através de mensagem enviada para objetos de alguma de suas subclasses, por exemplo classe Fonte. A figura 4.1 apresenta as classes que compõem esta primeira hierarquia.

Durante o desenvolvimento do SIM/SAVAD foram definidas diversas listas, por exemplo, uma para armazenar os nodos que são criados, outras para armazenar os clientes que estão num nodo. Estas listas foram organizadas numa hierarquia, onde a superclasse define métodos que serão utilizados pelas listas associadas as subclasses. A figura 4.2 apresenta as classes que compõem esta hierarquia.

Outras hierarquias que foram definidas representam os geradores de amostras, as disciplinas de escalonamento de fila do nodo Ponto Escalonador e os tipos de fila. As figuras 4.3, 4.4 e 4.5 apresentam, respectivamente, as classes que compõem essas hierarquias.

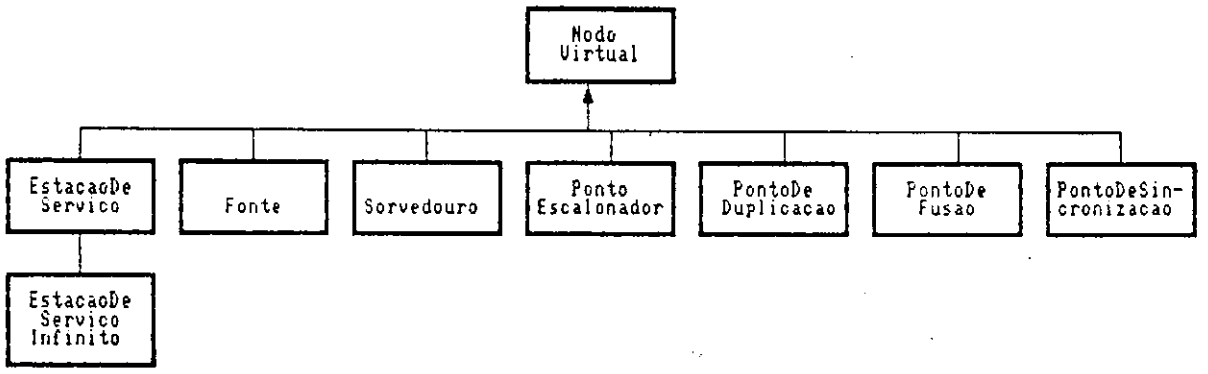


Figura 4.1 - Hierarquia de Classes para a Representacao de Nodos

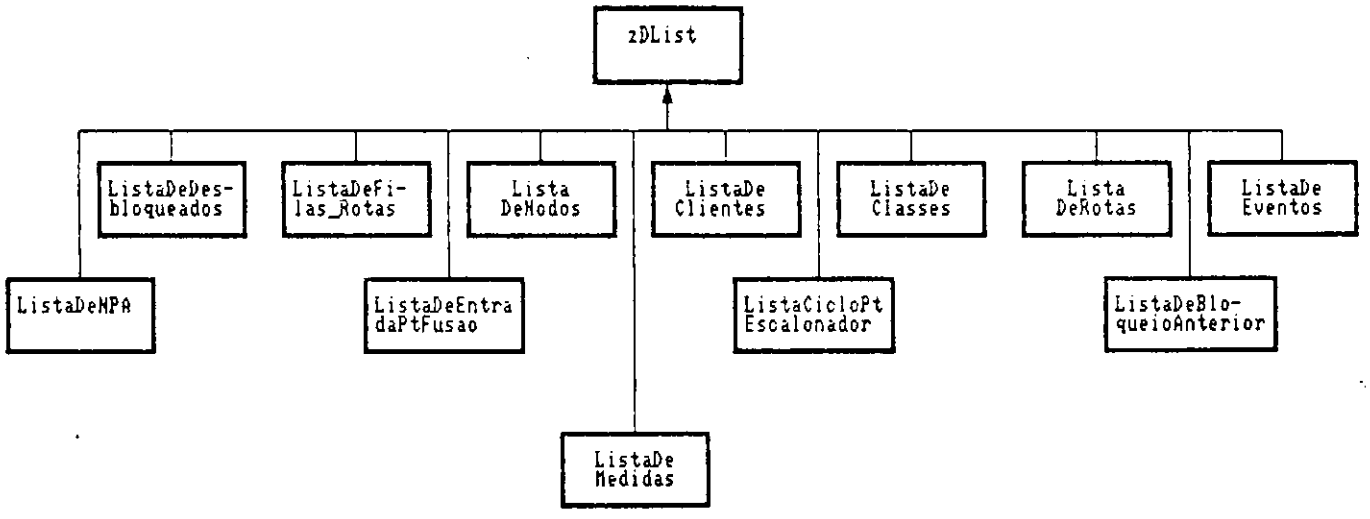


Figura 4.2 - Hierarquia de Classes para a Representacao de Listas

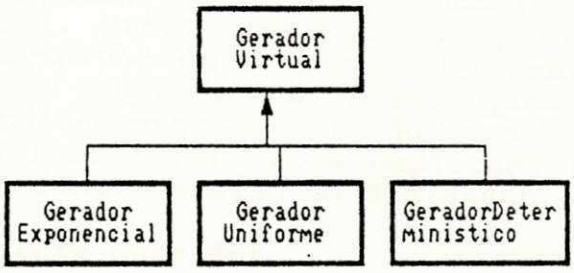


Figura 4.3 - Hierarquia de Classes para a Representacao de Geradores de Amostras

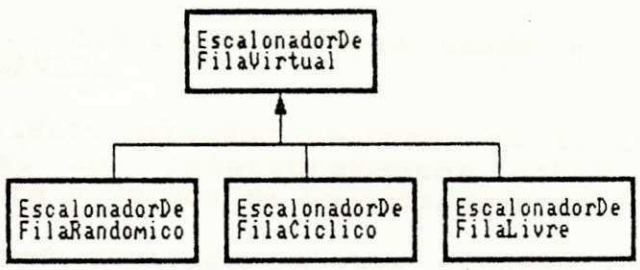


Figura 4.4 - Hierarquia de Classes para a Representacao de Escalonadores de Fila

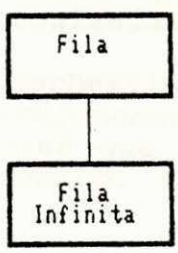


Figura 4.5 - Hierarquia de Classes para a Representacao de Filas

Durante o projeto do SIM/SAVAD foram exploradas as potencialidades do polimorfismo e dos mecanismos de herança e de amarração dinâmica. Estes recursos do paradigma de objetos facilitaram a definição da sequência de mensagens para quatro grupos de classes, a saber: nodos, filas, geradores de amostras e disciplinas de escalonamento de fila do nodo Ponto Escalonador. A utilização dessas facilidades será explicada na Seção 4.4, que aborda aspectos da implementação do SIM/SAVAD.

4.2.2 - Modelo Cliente-Servidor

O modelo cliente-servidor [Souto 92] fornece uma visão geral do relacionamento entre os objetos. Nesse modelo cada objeto interage com os outros objetos através de mensagens que passam ou buscam informações, solicitam aos objetos a implementação de um procedimento, etc. [Dias 92].

A figura 4.6 mostra a notação usada no modelo cliente-servidor do SIM/SAVAD.

O modelo cliente-servidor mostrado na figura 4.7a oferece uma visão global dos relacionamentos entre as classes do SIM/SAVAD. A figura 4.7b mostra as mensagens desse modelo.

Os subsistemas mostrados nessas figuras e detalhados nas demais figuras desta Subseção (4.8 a 4.14), correspondem as cinco hierarquias de classes que foram definidas para o SIM/SAVAD.

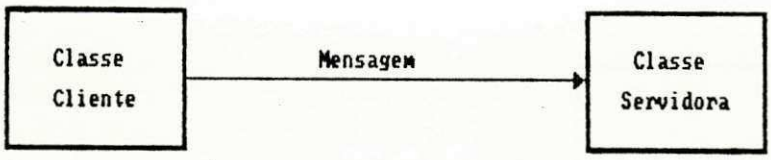
O subsistema Nodos agrega as classes que representam os tipos de nodos do SIM/SAVAD, isto é, estação de serviço, fonte, sorvedouro, ponto de fusão, ponto escalonador, ponto de duplicação e ponto de sincronização.

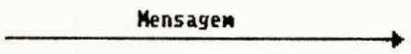
O subsistema EscalonadorDeFila representa as disciplinas que podem ser utilizadas no escalonamento de filas nos nodos do tipo Ponto Escalonador. As classes que esse subsistema agrega são as seguintes: EscalonadorRandômico, EscalonadorCíclico e EscalonadorLivre.

O subsistema GeradorDeAmostras representa a geração de amostras para as variáveis aleatórias dos modelos do SIM/SAVAD. As classes que esse subsistema agrega são as seguintes: GeradorDeAmostrasExponencial, GeradorDeAmostrasUniforme, e GeradorDeAmostrasDeterministico.

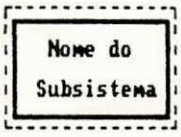
O subsistema Listas agrega o conjunto de classes do tipo listas, a saber: ListaDeRotas, ListaDeEntradaPtFusao, ListaCicloPtEscalonador, ListaDeFilas_Rotas, ListaDeBloqueioAnterior, ListaDeClientes, ListaDeDesbloqueio, ListaDeNodos, ListaDeEventos, ListaDeNPA, ListaDeClasses e ListaDeMedidas.

O subsistema Filas agrega as classes Fila e FilaInfinita.





: representa o envio de mensagem(s) do objeto da classe cliente para o objeto da classe servidora



: conjunto de classes de uma hierarquia

: expansao de um subsistema apresentando suas classes componentes

Figura 4.6 - Notacao do Modelo Cliente-Servidor

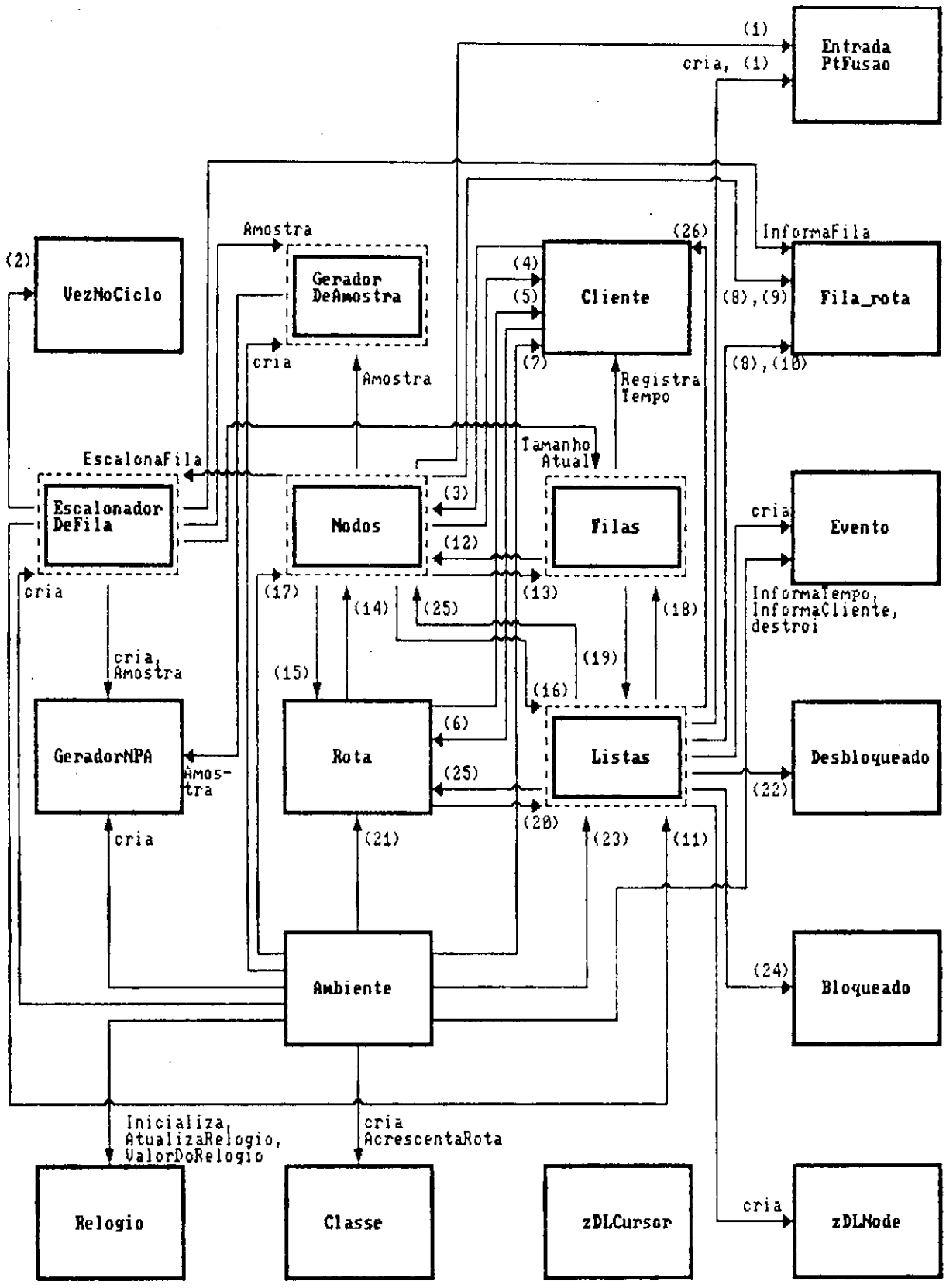


Figura 4.7a - Modelo Cliente-Servidor

(1) InformaRota, InformaListaDeBloqueioAnterior

(2) InformaNumDaFila, InformaNumDeLiberacoes, InformaFlag, oria

(3) DizTipo, VisitaNo, TestaDesbloqueioEmNoAnterior, Saida

(4) oria, RegistraTempo, RegistraNascimento, RegistraNo, EliminaCliente, InformaRota

(5) RegistraNo

(6) ContaSaida

(7) InicializaClass, Avanco, InformaTempo, InformaRota, InformaNodoAtual, RelataEstatistica

(8) InformaFila, InformaVflagDeDisponibilidade, InformaVrota

(9) AtualizaVflagDeDisponibilidade

(10) oria, RegistraRota

(11) AchaFilaPeloNumero, InformaVezNoCiclo

(12) DizTipo, TrataDesbloqueioDeEmergencia

(13) oria, InformaSeVazia, Visita, InsereClienteBloqueado, IndicaPrimeiroCliente, PegaDaFila, TestaDesbloqueioEmNoAnterior, TamanhoAtual, RetiraAteNClientes, Disponibilidade

(14) DizTipo, EstadoDisponivel, Disponibilidade, VisitaNo

(15) PosicionaNClientes, RetornaProxNo

(16) oria, SatisfazNovaFusao, CondicaoSatisfeita, SatisfazCondicaoComRota, InformaEntrada, AchaEntradaPelaRota, InformaSeVazia, InformaBloqueado, RetiraBloqueado, InsereBloqueado, AchaFilaPeloNumero, AchaFilaPelaRota, AchaNumDaFilaPelaFila, Visita, RetiraCliente, InsereCliente, InsereDesbloqueado, InsereEvento, InsereRota

(17) oria, InicializaFonte, DizTipo, VisitaNo, TrataClienteBloqueado, InsereClienteBloqueado, TrataDesbloqueio, TestaDesbloqueioEmNoAnterior

(18) Visita, InsereClienteBloqueado

(19) oria, InformaSeVazia, Informa Bloqueado, RetiraBloqueado, InsereBloqueado, RetiraCliente, InsereCliente

(20) GeraNClientes, RetiraCliente, InsereDesbloqueado, InformaNo

(21) oria, Inicializa, ProximoNo, DizTipo, PrimeiroNo, IniciCFechada

(22) oria, InformaCliente, InformaNo, destroi

(23) oria, InsereNodo, InsereClasse, InsereGeradorNPA, InsereRota, PegaDesbloqueio, PegaCliente, RemoveEvento, destroi, RelataEstatistica

(24) oria, InformaTipo, InformaNumDeClientesSaindo, InformaBloqueado

(25) RelataEstatistica

(26) InformaRota, oria

Figura 4.7b - Mensagens do Modelo Cliente-Servidor

No modelo cliente-servidor do SIM/SAVAD deve-se perceber o papel desempenhado pela classe Ambiente. Segundo [Takahashi 90:77] "... um programa será composto por uma classe cuja instanciação será iniciada pelo programador (caracterizando uma espécie de programa principal) ...", para cumprir esta função foi definida a classe Ambiente.

Os principais métodos da classe Ambiente são discutidos na Subseção 4.2.3.

Como o modelo da figura 4.7a apresenta um número grande de relacionamentos, um melhor detalhamento deste modelo pode ser obtido através de diagramas que apresentam os relacionamentos (mensagens enviadas/recebidas) dos subsistemas com as demais classes do SIM/SAVAD.

As figura 4.8a e 4.8b apresentam o diagrama das mensagens recebidas pelas classes do subsistema Nodos provenientes de outras classes do SIM/SAVAD.

Fazendo parte do subsistema Nodos encontra-se o subsistema Estacao que representa as estações de serviço (servidores simples, servidores múltiplos e servidores infinitos).

A figura 4.9a apresenta o diagrama das mensagens enviadas pelas classes do subsistema Nodos que se destinam às outras classes do SIM/SAVAD. A figura 4.9b apresenta as mensagem desse diagrama.

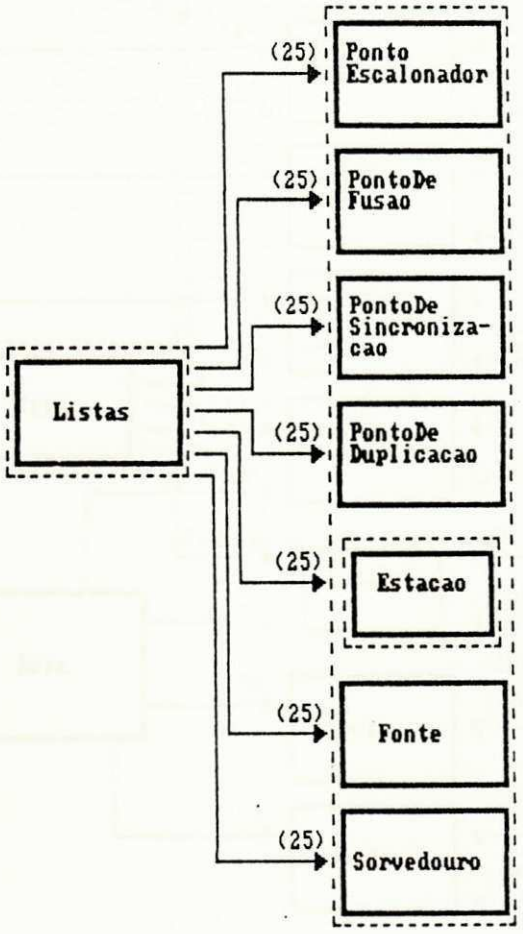
A figura 4.10a apresenta o diagrama das mensagens recebidas/enviadas pelas classes do subsistema Listas das/para as demais classes do SIM/SAVAD. Esta troca de mensagens pode ser melhor entendida a partir da descrição de cada classe, conforme os protocolos que se encontram no apêndice A. A figura 4.10b apresenta as mensagens desse diagrama.

A figura 4.11 apresenta o diagrama das mensagens recebidas/enviadas pelas classes do subsistema EscalonadorDeFila das/para as demais classes do SIM/SAVAD.

A figura 4.12 apresenta o diagrama das mensagens recebidas/enviadas pelas classes do subsistema GeradorDeAmostras das/para as demais classes do SIM/SAVAD.

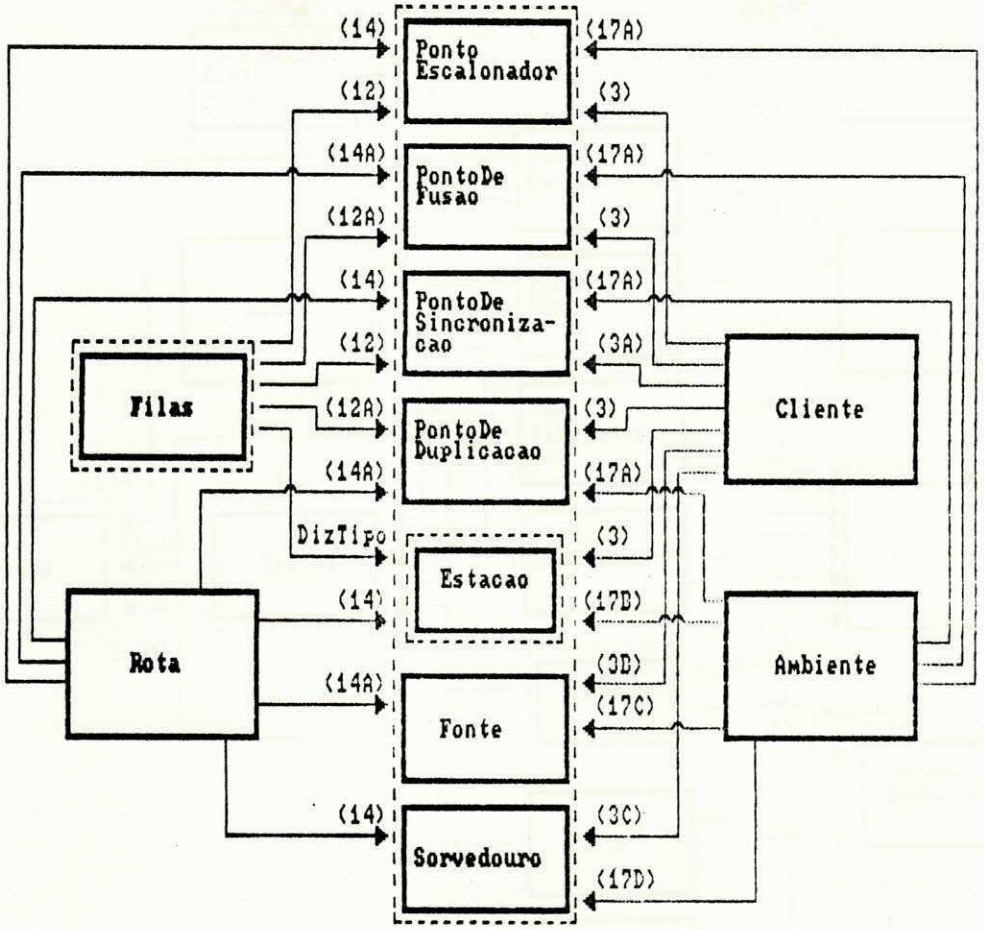
A figura 4.13 apresenta o diagrama das mensagens recebidas/enviadas pelas classes do subsistema Filas das/para as demais classes do SIM/SAVAD.

A figura 4.14 apresenta o diagrama das mensagens recebidas/enviadas pelas classes do subsistema Estacao das/para as demais classes do SIM/SAVAD.



(25) RelataEstatistica

Figura 4.8a - Diagrama de Mensagens Recebidas pela Expansao do Subsistema Nodos



(3) DizTipo, VisitaNo, TestaDesbloqueioEmNoAnterior, Saida

(3A) DizTipo, VisitaNo

(3B) DizTipo, TestaDesbloqueioEmNoAnterior, Saida

(3C) DizTipo, VisitaNo, TestaDesbloqueioEmNoAnterior

(12) DizTipo, TrataDesbloqueioDeEmergencia

(12A) TrataDesbloqueioDeEmergencia

(14) DizTipo, EstadoDisponivel, Disponibilidade, VisitaNo

(14A) DizTipo

(17A) cria, DizTipo, VisitaNo, TrataClienteBloqueado, InsereClienteBloqueado, TestaDesbloqueioEmNoAnterior

(17B) cria, DizTipo, VisitaNo, TrataClienteBloqueado, InsereClienteBloqueado, TrataDesbloqueio, TestaDesbloqueioEmNoAnterior

(17C) cria, InicializaFonte, DizTipo, TrataClienteBloqueado

(17D) cria, DizTipo, VisitaNo

Figura 4.8b - Mensagens Recebidas pela Expansao do Subsistema Nodos provenientes de outros Subsistemas

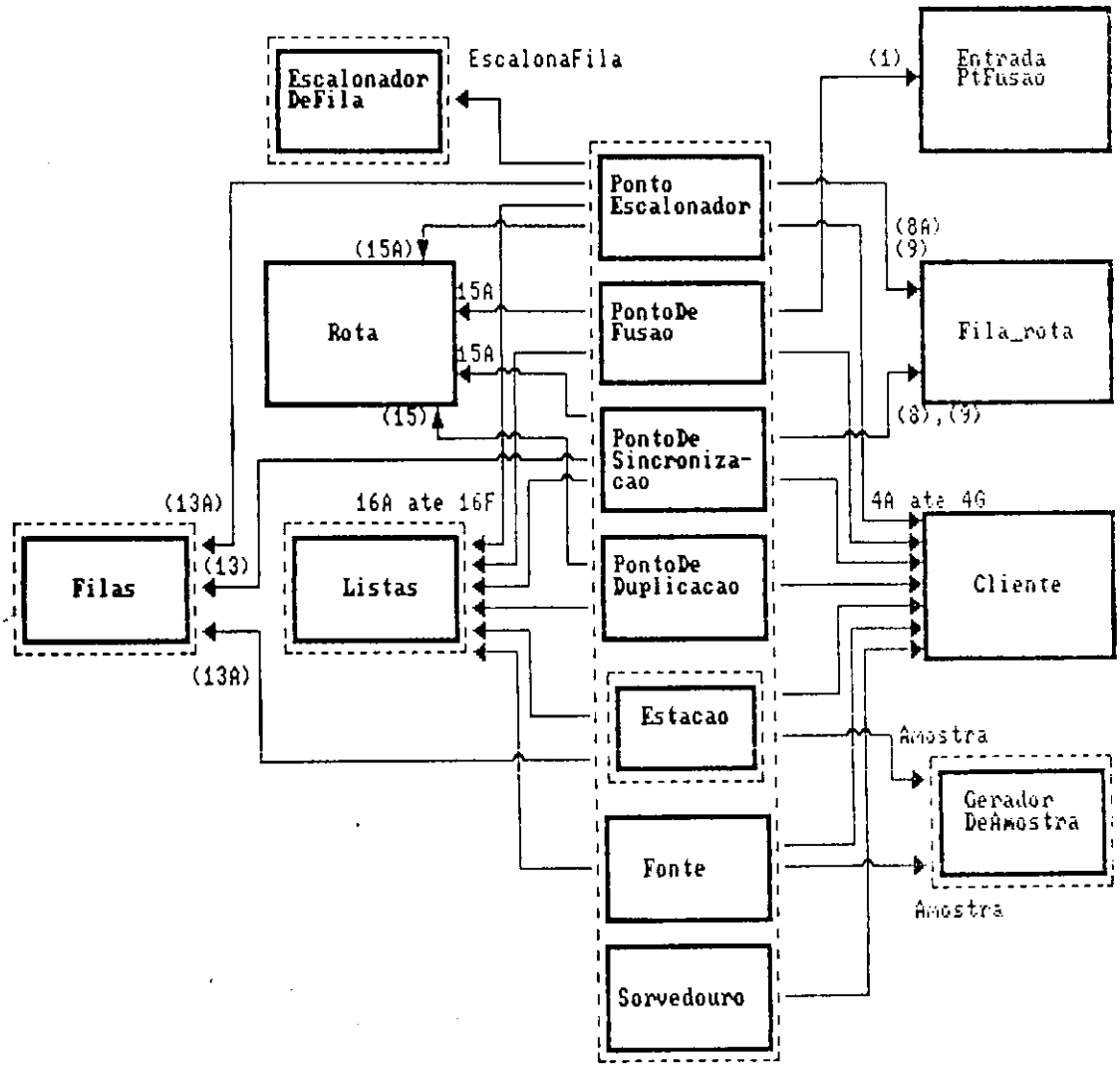


Figura 4.9a - Diagrama de Mensagens Enviadas pela Expansao do Subsistema Nodos

(1) InformaRota, InformaListaDeBloqueioAnterior

(4A) Mensagens da classe PontoEscalonador para classe Cliente
RegistraNo, InformaRota

(4B) Mensageens da classe PontoDeFusao para classe Cliente
EliminaCliente, InformaRota

(4C) Mensagens da classe PontoDeSincronizacao para classe Cliente
RegistraNo, EliminaCliente, InformaRota

(4D) Mensagens da classe PontoDeDuplicacao para classe Cliente
cria, RegistraNascimento

(4E) Mensagens do subsistema Estacao para classe Cliente
RegistraTempo, RegistraNo

(4F) Mensagens da classe Fonte para classe Cliente
cria, RegistraTempo, RegistraNascimento

(4G) Mensagem da classe Sorvedouro para classe Cliente
EliminaCliente

(8) InformaFila, InformaUflagDeDisponibilidade, InformaUrota

(8A) InformaFila, InformaUflagDeDisponibilidade

(9) AtualizaUflagDeDisponibilidade

(13) cria, InformaSeVazia, Visita, InsereClienteBloqueado, IndicaPrimeiroCliente,
PegaDaFila, TestaDesbloqueioEmNoAnterior, TamanhoAtual, RetiraAteNClientes, Disponibilidade

(13A) cria, InformaSeVazia, Visita, InsereClienteBloqueado, IndicaPrimeiroCliente,
PegaDaFila, TestaDesbloqueioEmNoAnterior, TamanhoAtual, Disponibilidade

(15) PosicionaNClientes, RetornaProxNo

(15A) RetornaProxNo

(16A) Mensagens da classe PontoEscalonador para classe Listas
cria, RegistraRota, Visita, InsereEvento, AchaFilaPeloNumero, AchaFilaPelaRota
AchaFilaPeloNumero, AchaFilaPelaRota

(16B) Mensagens da classe PontoDeFusao para classe Listas
cria, SatisfazNovaFusao, CondicaoSatisfeita, SatisfazCondicaoComRota, InformaEntrada,
AchaEntradaPelaRota, InformaBloqueado, RetiraBloqueado, InsereBloqueado,
InsereCliente, InsereDesbloqueado, InsereEvento, RetiraCliente,

(16C) Mensagens da classe PontoDeSincronizacao para classe Listas
AchaFilaPeloNumero, AchaFilaPelaRota, AchaNumDaFilaPelaFila, Visita, RetiraCliente

(16D) Mensagens da classe PontoDeDuplicacao para classe Listas
cria, InformaSeVazia, InformaBloqueado, RetiraBloqueado, InsereBloqueado
InsereDesbloqueado, InsereEvento,

(16E) Mensagem do subsistema Estacao para classe Listas
InsereEvento

(16F) Mensagem da classe Fonte para classe Listas
InsereEvento

Figura 4.9b - Mensagens Enviadas pela Expansao do Subsistema Nodos

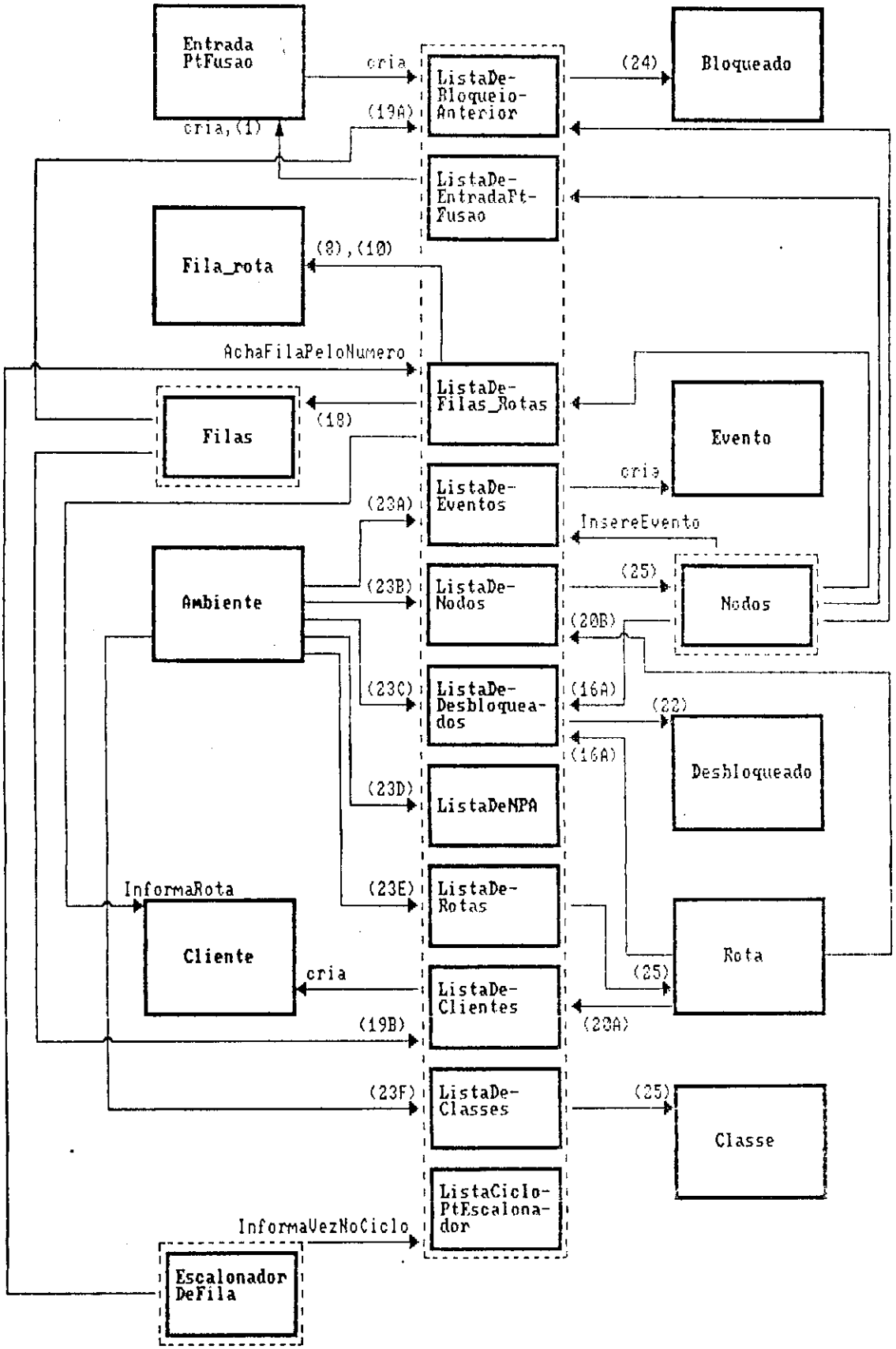


Figura 4.10a - Diagrama de Mensagens Recebidas/Enviadas pela Expansao do Subsistema Listas

(1) InformaRota, InformaListaDeBloqueioAnterior

(8) InformaFila, InformaUflagDeDisponibilidade, InformaUrota

(10) cria, RegistraRota

(16A) cria, SatisfazNovaFusao, CondicaoSatisfeita, SatisfazCondicaoComRota, InformaEntrada, AchaEntradaPelaRota

(16B) AchaFilaPeloNumero, AchaFilaPelaRota, AchaNumDaFilaPelaFila, Visita

(18) Visita, InsereClienteBloqueado

(19A) cria, InformaBloqueado, RetiraBloqueado, InsereBloqueado

(19B) cria, RetiraCliente, InsereCliente

(20A) GeraNClientes, RetiraCliente

(20B) InformaNo

(23A) cria, RemoveEvento, destroi

(23B) cria, InsereNodo, destroi, RelataEstatistica

(23C) cria, PegaDesbloqueio, PegaCliente, destroi

(23D) cria, InserGeradorNPA, destroi

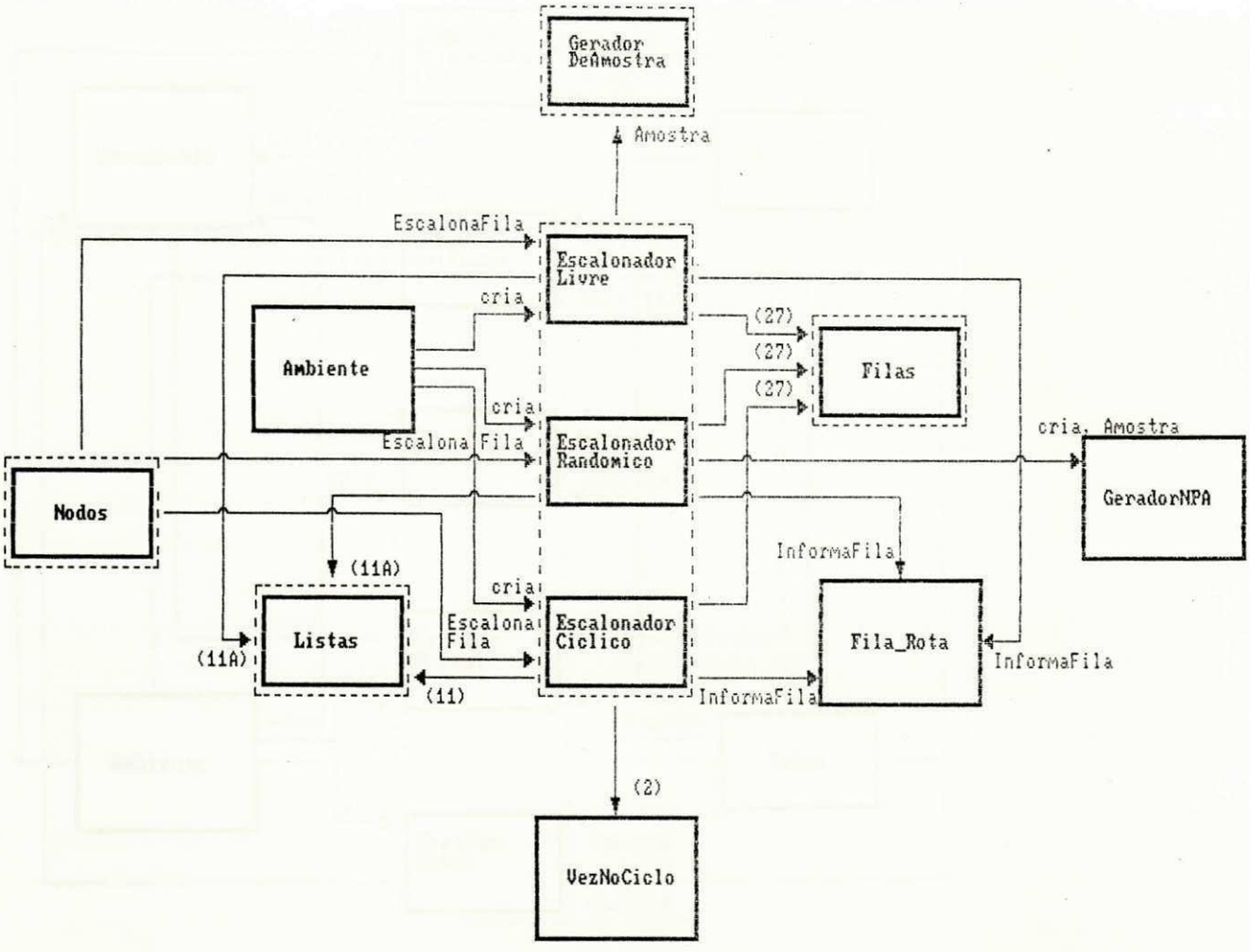
(23E) cria, InsereRota, RelataEstatistica, destroi

(23F) cria, InsereClasse, RelataEstatistica, destroi

(24) cria, InformaTipo, InformaNumDeClientesSaindo, InformaBloqueado

(25) RelataEstatistica

Figura 4.10b - Continuacao da Figura 4.10a



(2) InformaNumDaFila, InformaNumDeLiberacoes, InformaFlag, cria

(11) AchaFilaPeloNumero, InformaVezNoCiclo

(11A) AchaFilaPeloNumero

(27) TamanhoAtual

Figura 4.11 - Diagrama de Mensagens Recebidas/Enviadas pela Expansao do Subsistema EscalonadorDeFila

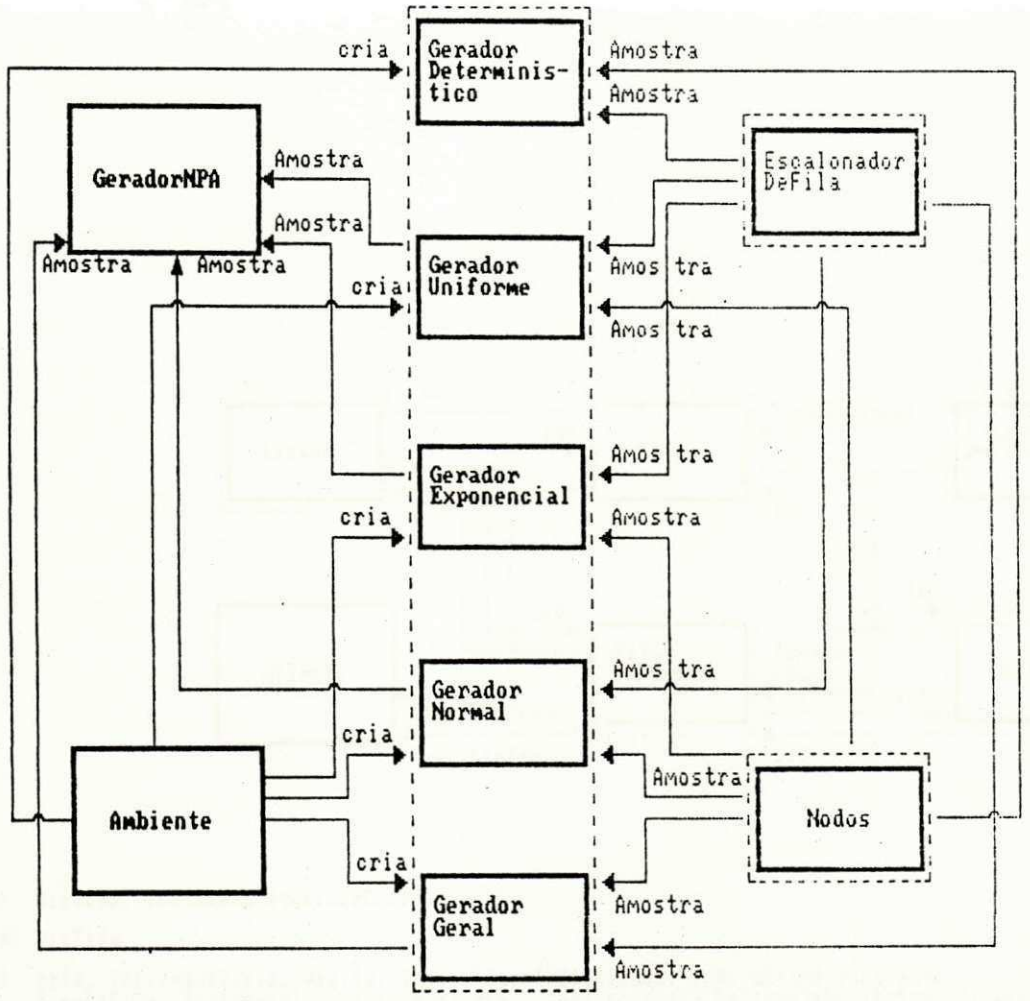
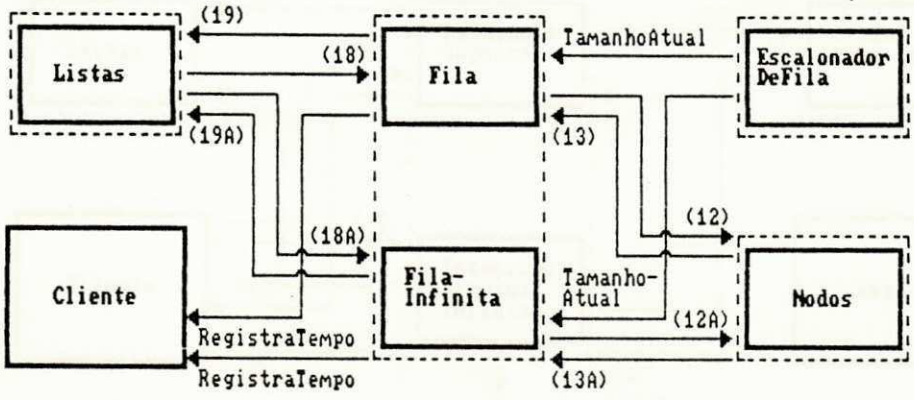


Figura 4.12 - Diagrama de Mensagens Recebidas/Enviadas pela Expansao do Subsistema GeradorDeAmostras



(12) DizTipo, TrataDesbloqueioDeEmergencia

(12A) DizTipo

(13) cria, InformaSeVazia, Visita, InsereClienteBloqueado, IndicaPrimeiroCliente, PegaDaFila, TestaDesbloqueioEmNoAnterior, TamanhoAtual, RetiraAteNClientes, Disponibilidade

(13A) cria, InformaSeVazia, Visita, IndicaPrimeiroCliente, PegaDaFila, TamanhoAtual RetiraAteNClientes, Disponibilidade

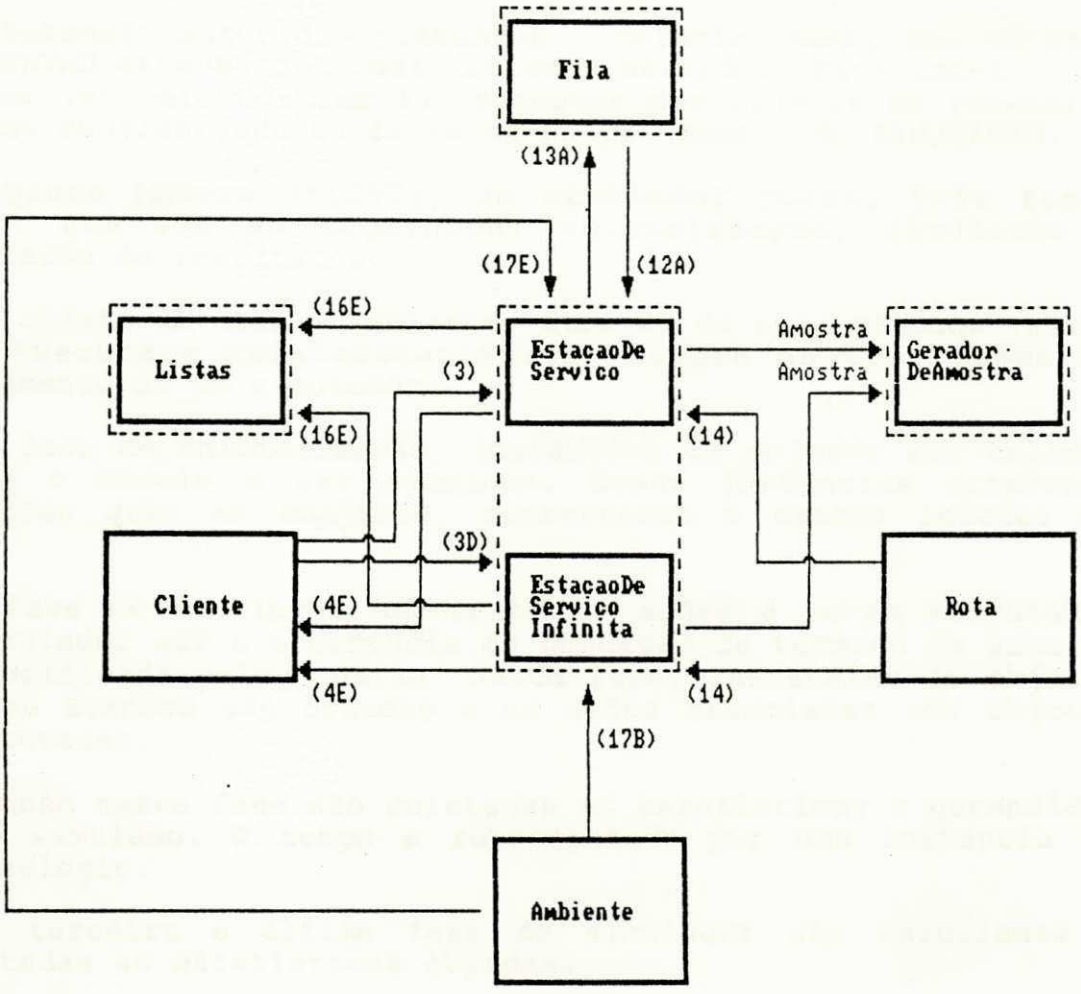
(18) Visita, InsereClienteBloqueado

(18A) Visita

(19) cria, InformaSeVazia, Informa Bloqueado, RetiraBloqueado, InsereBloqueado, RetiraCliente, InsereCliente

(19A) cria, RetiraCliente, InsereCliente

Figura 4.13 - Diagrama de Mensagens Recebidas/Enviadas pela Expansao do Subsistema Filas



(3) DizTipo, VisitaNo, TestaDesbloqueioEmNoAnterior, Saida

(3D) DizTipo, VisitaNo, Saida

(4E) RegistraTempo, RegistraNo

(12A) DizTipo

(13A) cria, InformaSeVazia, Visita, InsereClienteBloqueado, IndicaPrimeiroCliente, PegaDaFila, TestaDesbloqueioEmNoAnterior, TamanhoAtual, Disponibilidade

(14) DizTipo, EstadoDisponivel, Disponibilidade, VisitaNo

(16E) InsereEvento

(17B) cria, DizTipo, VisitaNo, TrataClienteBloqueado, InsereClienteBloqueado, TrataDesbloqueio, TestaDesbloqueioEmNoAnterior

(17E) cria, DizTipo, VisitaNo, TrataClienteBloqueado, TrataDesbloqueio

Figura 4.14 - Diagrama de Mensagens Recebidas/Enviadas pela Expansao do Subsistema Estacao

4.2.3 - Extensão do Modelo Cliente-Servidor Para as Fases do SIM/SAVAD

A Subseção anterior apresentou o relacionamento dos objetos do SIM/SAVAD através do modelo cliente-servidor. Este modelo pode ser estendido para mostrar as respostas dos objetos ao receberem mensagens considerando as fases de processamento do SIM/SAVAD.

Segundo [Moura 86:297], um simulador possui três fases básicas, que são as seguintes: inicialização, simulação e apresentação de resultados.

Um objeto da classe Ambiente, através de seus métodos Inicializa, Executa e RelataEstatistica, cumpre as três fases de processamento de um simulador.

Na fase de inicialização, instâncias de objetos são criadas conforme o modelo a ser simulado. Essas instâncias armazenam informações que, em conjunto, representam o estado inicial do modelo.

A fase de simulação apresenta as ações a serem executadas pelo simulador até a ocorrência da condição de término de simulação especificada pelo usuário. Nesta fase, instâncias de objetos da classe Eventos são criadas e as ações associadas aos objetos são executadas.

Também nessa fase são coletadas as estatísticas e gerenciado o tempo simulado. O tempo é representado por uma instância da classe Relógio.

Na terceira e última fase do simulador são calculadas e apresentadas as estatísticas obtidas.

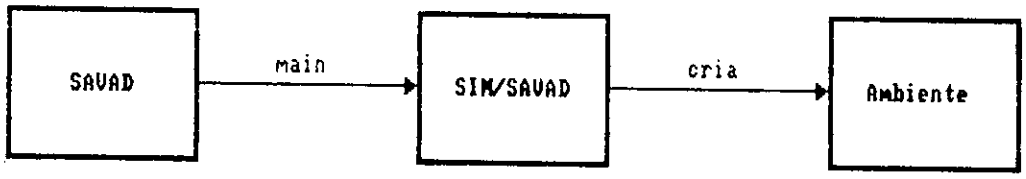
A figura 4.15a apresenta o modelo cliente-servidor para a fase de inicialização. A figura 4.15b é uma continuação da figura 4.15a.

Essa fase começa com a ativação do SIM/SAVAD pelo SAVAD. Uma instância da classe Ambiente é criada e o método Inicializa é ativado.

O método Inicializa desencadeia o processo de criação de instâncias das classes mostradas no diagrama cliente-servidor.

Ainda nesta fase são criadas instâncias das classes Evento e Cliente, necessárias ao início da fase de simulação.

(a) Ativacao do SIM/SAVAD e criacao de objeto da classe Ambiente



(b) SIM/SAVAD envia mensagem Inicializa para objeto da classe Ambiente

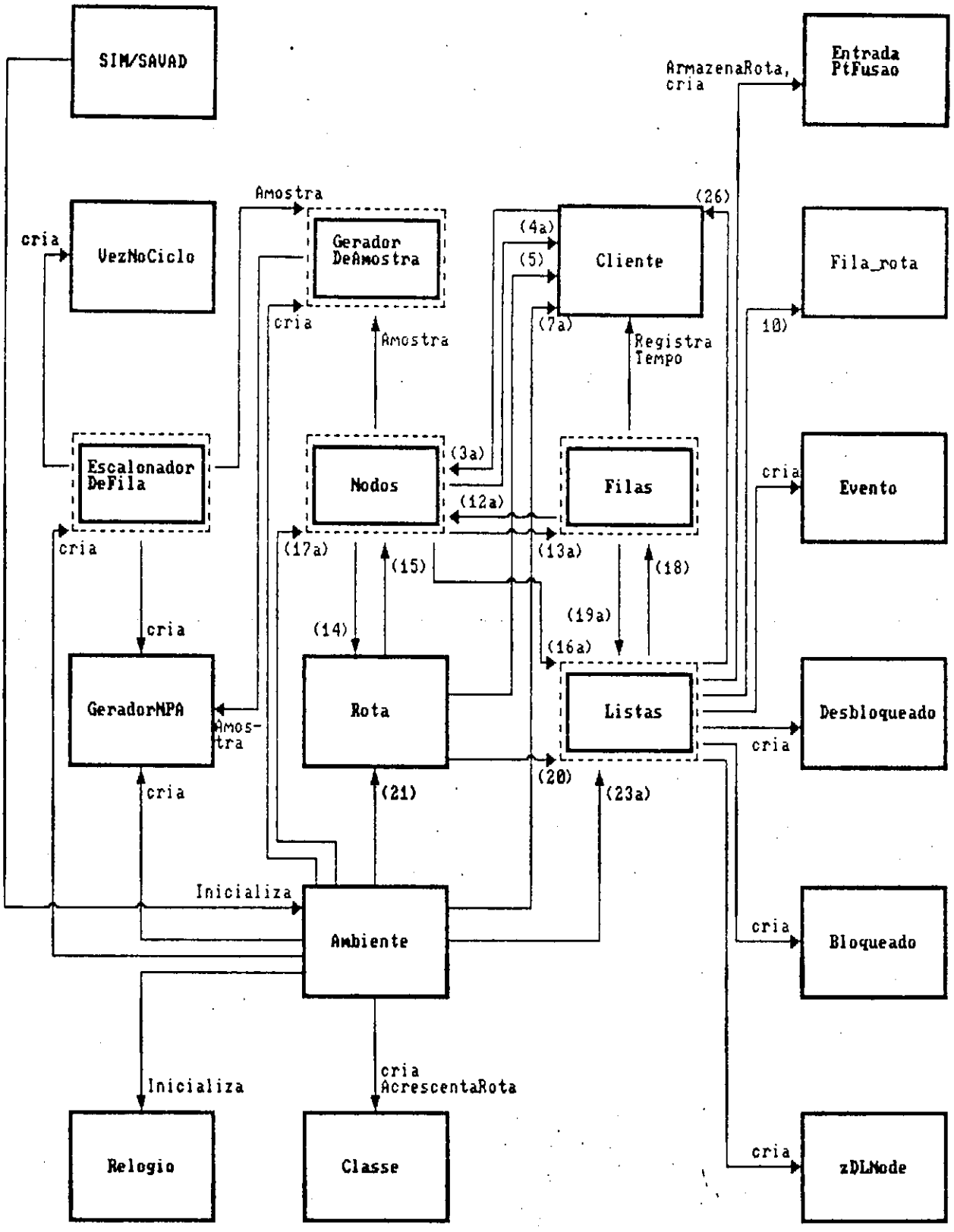


Figura 4.15a - Mensagens trocadas apos o envio da mensagem Inicializa para objeto da classe Ambiente

Mensagens utilizadas entre o envio da mensagem Inicializa e da mensagem Executa para a instancia da classe Ambiente que foi criada pelo SIM/SAVAD

(3a) DizTipo, VisitaNo, TestaDesbloqueioEmNoAnterior

(4a) cria, RegistraTempo, RegistraNascimento, RegistraNo, InformaRota

(5) RegistraNo

(7a) InicializaClass

(10) cria, RegistraRota

(12a) DizTipo

(13a) cria, Visita, InsereClienteBloqueado

(14) DizTipo, EstadoDisponivel, Disponibilidade, VisitaNo

(15) PosicionaNClientes, RetornaProxNo

(16a) cria, AchaEntradaPelaRota, InsereBloqueado, RegistraRota, AchaFilaPeloNumero, AchaFilaPelaRota, Visita, RetiraCliente, InsereCliente, InsereDesbloqueado, InsereEvento, InsereRota

(17a) cria, InicializaFonte, DizTipo, VisitaNo, TrataClienteBloqueado, InsereClienteBloqueado,

(18) Visita, InsereClienteBloqueado

(19a) cria, InsereBloqueado, InsereCliente

(20) GeraNClientes, RetiraCliente, InsereDesbloqueado, InformaNo

(21) cria, Inicializa, ProximoNo, DizTipo, PrimeiroNo, IniciCFechada

(23a) cria, InsereNodo, InsereClasse, InsereGeradorNPA, InsereRota

Figura 4.15b - Continuacao da Figura 4.15a

A fase de simulação começa com a ativação do método Executa da classe Ambiente e durante esta fase são escalonados eventos até que sejam satisfeitas as condições que limitam o término da simulação.

A figura 4.16a apresenta o modelo cliente-servidor para a fase de simulação. A figura 4.16b é uma continuação da figura 4.16a.

Quando for identificado o término da fase de simulação, o SIM/SAVAD ativa o método RelataEstatistica da classe Ambiente para começar a fase de apresentação de resultados.

Nesta fase são ativados os métodos RelataEstatistica das classes Cliente, ListaDeNodos, ListaDeClasses, ListaDeRotas, ListaDeEventos e ListaDeDesbloqueio. Esses métodos apresentam estatísticas sobre clientes, nodos, rotas e classes, conforme medidas de desempenho apresentadas na Seção 2.3.

A figura 4.17 apresenta o diagrama cliente-servidor correspondente a fase de apresentação de resultados.

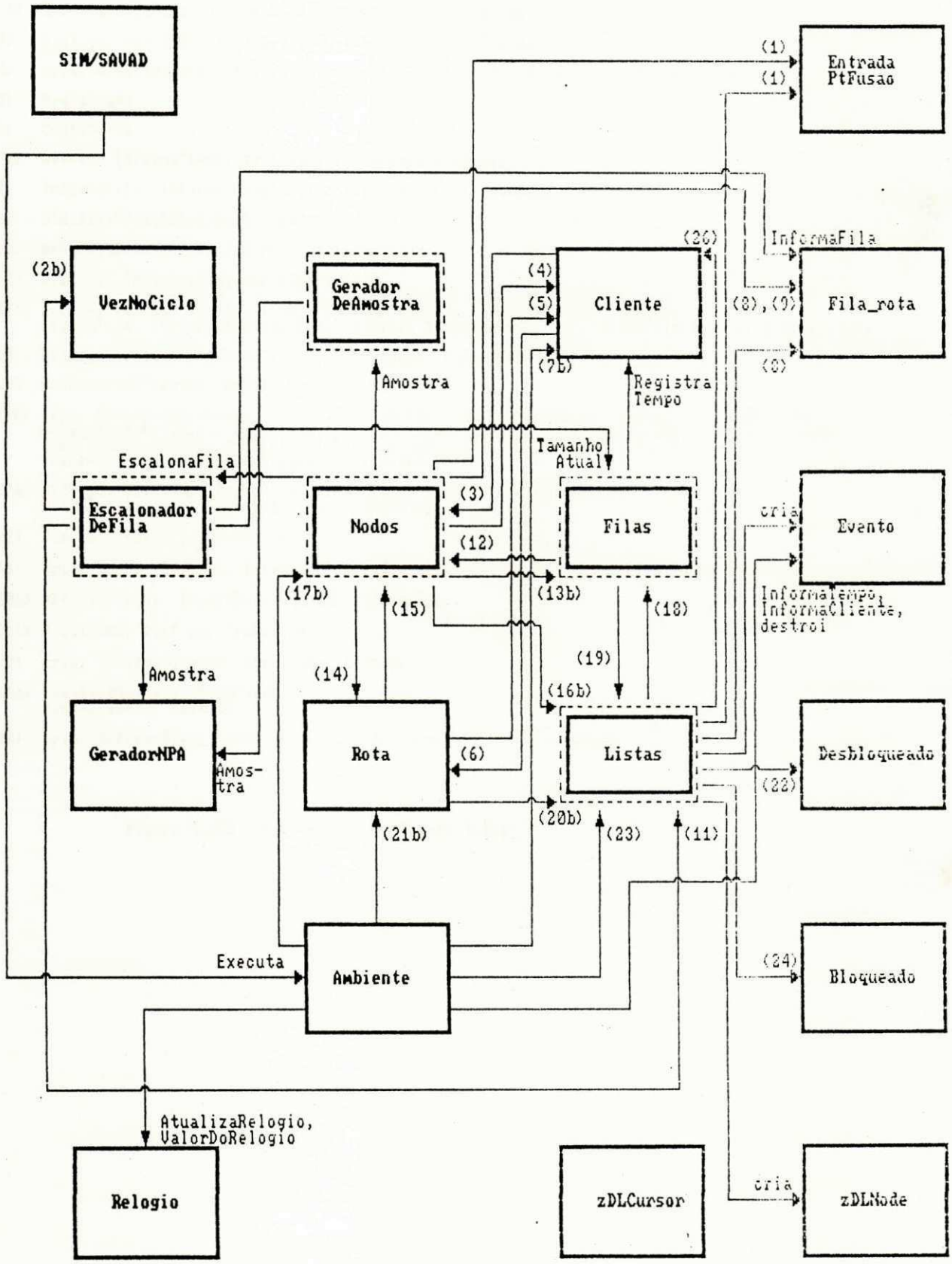


Figura 4.16a - Modelo Cliente-Servidor para a fase de Simulacao

Mensagens utilizadas entre o envio da mensagem Executa e da mensagem RelataEstatistica para a instancia da classe Ambiente que foi criada pelo SIM/SAVAD

(1) InformaRota, InformaListaDeBloqueioAnterior

(2b) InformaNumDaFila, InformaNumDeLiberacoes, InformaFlag

(3) DizTipo, VisitaNo, TestaDesbloqueioEmNoAnterior, Saida

(4) cria, RegistraTempo, RegistraNascimento, RegistraNo, EliminaCliente, InformaRota

(5) RegistraNo

(6) ContaSaida

(7b) Avanco, InformaTempo, InformaRota, InformaNodoAtual

(8) InformaFila, InformaVflagDeDisponibilidade, InformaVrota

(9) AtualizaVflagDeDisponibilidade

(11) AchaFilaPeloNumero, InformaVezNoCiclo

(12) DizTipo, TrataDesbloqueioDeEmergencia

(13b) InformaSeVazia, Visita, InsereClienteBloqueado, IndicaPrimeiroCliente, PegaDaFila, TestaDesbloqueioEmNoAnterior, TamanhoAtual, RetiraAteNClientes, Disponibilidade

(14) DizTipo, EstadoDisponivel, Disponibilidade, VisitaNo

(15) PosicionaNClientes, RetornaProxNo

(16b) cria, SatisfazNovaFusao, CondicaoSatisfeita, SatisfazCondicaoComRota, InformaEntrada, AchaEntradaPelaRota, InformaSeVazia, InformaBloqueado, RetiraBloqueado, InsereBloqueado, AchaFilaPeloNumero, AchaFilaPelaRota, AchaNumDaFilaPelaFila, Visita, RetiraCliente, InsereCliente, InsereDesbloqueado, InsereEvento

(17b) DizTipo, VisitaNo, TrataClienteBloqueado, InsereClienteBloqueado, TrataDesbloqueio, TestaDesbloqueioEmNoAnterior

(18) Visita, InsereClienteBloqueado

(19) cria, InformaSeVazia, Informa Bloqueado, RetiraBloqueado, InsereBloqueado, RetiraCliente, InsereCliente

(20b) RetiraCliente, InsereDesbloqueado, InformaNo

(21b) ProximoNo, DizTipo, PrimeiroNo

(22) cria, InformaCliente, InformaNo, destroi

(23b) PegaDesbloqueio, PegaCliente, RemoveEvento, destroi

(24) cria, InformaTipo, InformaNumDeClientesSaindo, InformaBloqueado

Figura 4.16b - Continuacao da Figura 4.16a

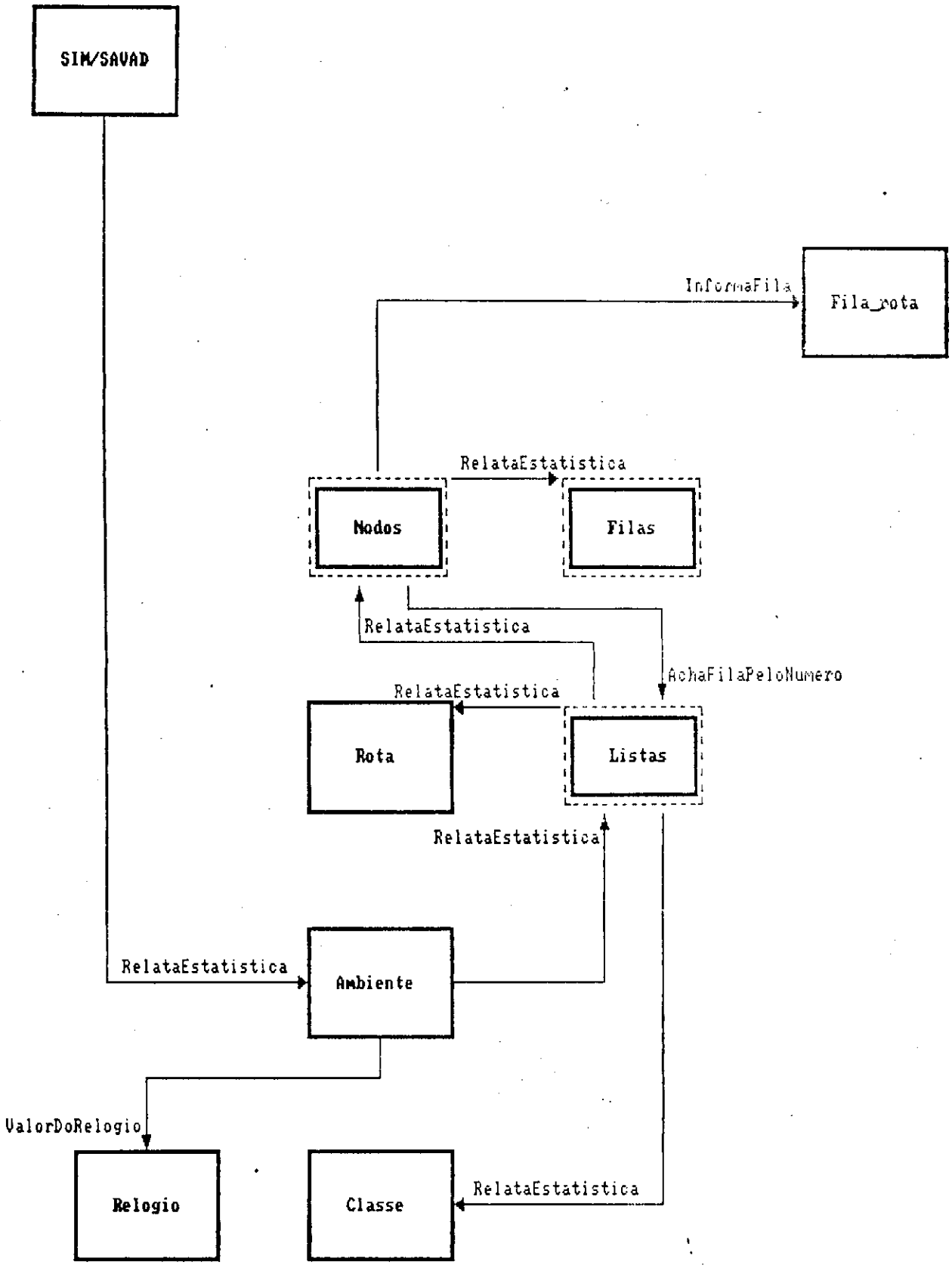


Figura 4.17 - Modelo Cliente-Servidor para a fase de Apresentacao de Resultados

4.3 - Comentários sobre Polimorfismo e Amarração Dinâmica

Como colocado em [Takahashi 90:150] "Em geral, abstrações interessantes são **descobertas** e não **inventadas**: o programador primeiro cria classes para resolver problemas específicos, e depois verifica que, com algumas modificações elas podem ser generalizadas para utilização mais ampla".

Uma das vantagens da escolha do paradigma de objetos foi inicialmente observada quando da estruturação de uma hierarquia de classes para representar os nodos do SIM/SAVAD. No início da fase de projeto desse simulador notou-se que cada tipo de nodo deveria pertencer a uma classe. Em seguida verificou-se que a definição de uma **classe abstrata** que fosse superclasse destas simplificaria o escalonamento de eventos e facilitaria a extensibilidade do SIM/SAVAD, conforme foi observado posteriormente na fase de implementação, que será discutido na próxima seção.

A classe abstrata definida recebeu o nome de NodoVirtual, e conforme [Takahashi 90:143] inclui "... métodos em termos de alguns outros métodos que deverão ser implementados nas subclasses". Tais métodos estão associados aos eventos identificados.

Em [Takahashi 90:311] uma hierarquia de classes semelhante é analisada sob o ponto de vista de programação usando apontadores. Veja o que apresenta essa referência sobre a classe **shape**, a qual teria as mesmas características hierarquicas da classe NodoVirtual: "Assim, se instanciarmos um apontador à sua classe **shape** e fizermos com que ele aponte uma instância de uma de suas subclasses, tudo funcionará como se este apontador fosse do tipo da subclasse."

No caso do SIM/SAVAD se um apontador é definido para a classe NodoVirtual e recebe o endereço de uma instância de uma das subclasses de NodoVirtual, este apontador poderá ativar funções da subclasse definidas através de polimorfismo, desde que tenham sido definidas como virtuais (conceito da linguagem C++) na classe NodoVirtual.

Devido às facilidade de herança, do polimorfismo e amarração dinâmica o método que controla a movimentação de clientes não precisa conhecer qual tipo de nodo está associado ao evento que está sendo processado. Isso foi implementado por meio de funções (métodos) virtuais da linguagem C++ [Zortech 90].

Estas facilidades também são utilizadas com instâncias da hierarquia de classes para filas, geradores de amostra e para escalonadores de fila do nodo ponto escalonador.

Assim, um método que tenha acesso a uma dada fila, não necessita conhecer se a fila tem ou não capacidade de armazenamento limitada. Da mesma forma, um método que tenha acesso a um gerador de amostras não necessita conhecer qual a

distribuição de probabilidade associada a este gerador.

No caso de um ponto escalonador, um método pode usar uma instância da hierarquia de classe que representa escalonadores de fila, sem se preocupar em conhecer qual tipo de escalonamento ela implementa.

4.4 - Implementação

A implementação do SIM/SAVAD foi feita em ambiente MS-DOS e a linguagem escolhida foi C++ [Zortech 90], a qual suporta o paradigma de objetos. A linguagem C++ é uma extensão da linguagem procedural C [Kernigham 78], possuindo construções sintáticas que suportam o conceito de classes, de herança, polimorfismo e amarração dinâmica.

Segundo [Doyle 91], a principal característica a ser analisada na escolha de uma linguagem de programação para construção de programas de simulação é a velocidade de execução, pelo fato de programas de simulação usarem, tradicionalmente, uma quantidade muito grande do tempo do computador. O tempo de execução pode limitar a quantidade de análises que são realizadas com o programa. Doyle recomenda o uso da linguagem C++ devido ao seu bom desempenho e as vantagens das linguagens orientadas a objetos.

O polimorfismo e a amarração dinâmica utilizada no SIM/SAVAD foram projetados (e implementados) tendo em vista o uso de funções virtuais da linguagem C++ [Zortech 90]. Segundo [Pinson 91:13]: "Existem soluções de problema em que o vínculo tardio* é parte essencial de um bom projeto orientado para o objeto".

Ainda segundo [Pinson 91:203], numa linguagem como C++ "... a responsabilidade de determinar qual a versão apropriada de função (método) que será chamada é do usuário do software. Normalmente, uma instrução ramificada com vários desvios condicionais, como a instrução **switch** ou a instrução **if else** em C ou a instrução **CASE** em Modula-2, é usada para obter-se este controle (isto é, **if (o objeto que recebe a mensagem é este) faça_isto, else if (o objeto que recebe a mensagem é outro) faça_outra_coisa) else if ...**). Estas instruções ramificadas são replicadas várias vezes na aplicação do usuário. A manutenção fica complicada quando há necessidade de acrescentar-se comportamentos adicionais em tais programas. Cada instância de uma instrução com vários desvios precisa ser ampliada para incluir opções adicionais acrescentadas ao comportamanto do sistema. Estas modificações no código do usuário precisam ser feitas em muitos lugares do software."

A linguagem C++ permite acrescentar comportamento adicional a um programa, apenas acrescentando-se uma ou mais subclasses.

---

* - vínculo tardio é a tradução do autor para "dynamic binding"

Isto foi verificado, por exemplo, no caso da definição dos nodos pois se está trabalhando com apontadores para objetos da classe NodoVirtual, os quais recebem valor a partir dos métodos Cliente::InformaNodoAtual e Rota::ProximoNodo.

Para os apontadores da classe NodoVirtual são enviadas mensagens para métodos virtuais. Estes métodos estão associados com métodos definidos nas subclasses da classe NodoVirtual, que são ativados em tempo de execução.

Destacam-se oito métodos virtuais na classe NodoVirtual: TrataDesbloqueio, Saida, VisitaNo, TrataClienteBloqueado, InsereDesbloqueio, TestaDesbloqueioEmNoAnterior, EstadoDisponivel e RelataEstatistica.

Outro ponto importante é que ativação de métodos das subclasses da classe NodoVirtual só são feitas, diretamente, pelo método CriaElementos da classe Ambiente, no momento de geração de objetos da classe de nodo desejado.

Como resultado da implementação, as definições do conjunto de classes foram mapeadas para um código em C++ de aproximadamente 340 Kbytes, excetuando-se o código da interface.

CAPITULO 5

VALIDACAO

C A P I T U L O 5

VALIDAÇÃO

Neste Capítulo é mostrado o processo de validação do SIM/SAVAD. Nas Seções que seguem são apresentados resultados de modelos de redes de filas simulados pelo SIM/SAVAD. Estes resultados são comparados com resultados obtidos através de técnicas analíticas, baseadas na Teoria das Filas, e, em resultados obtidos através da execução de outros simuladores, nos casos em que soluções analíticas não podem ou são difíceis de ser aplicadas.

Para evitar os efeitos do transitório inicial sobre os resultados produzidos pelo SIM/SAVAD, utilizou-se uma das sugestões contidas em [Moura 86:300], a saber: "... executar o simulador por um tempo (simulado) grande bastante para tornar o transitório desprezível ...".

5.1 Validação através de Técnicas Analíticas

Esta seção compara resultados de simulação de sistemas obtidos com o SIM/SAVAD, com aqueles obtidos através da utilização de técnicas analíticas baseada em Teoria das Filas. Este conjunto abrange sistemas de filas associados a nomenclatura A/B/m/K/M e sistemas de redes de filas que podem ser solucionadas pelo teorema de Burke (redes de filas abertas), resultado BCMP e Análise do Valor Médio (redes de filas fechadas) e por Cadeias de Markov (redes de filas estendidas). Informações detalhadas sobre essas técnicas podem ser encontradas em [Moura 86].

A seguir apresentam-se os sistemas considerados na validação do SIM/SAVAD.

5.1.1 - Sistemas A/B/m/K/M

5.1.1.1- Sistema M/M/1

Este sistema é caracterizado pelas seguintes propriedades:

a) o processo de chegada de clientes apresenta distribuição exponencial com média 1/Ä. Esse processo é representado por uma fonte.

b) o processo de atendimento a solicitação de serviço dos clientes apresenta distribuição exponencial com média 1/ß. Esse

processo é representado por uma estação de serviço com servidor único.

c) Os clientes partem do sistema após serem atendidos na sua solicitação de serviço. Esse processo é representado por um sorvedouro.

O sistema M/M/1 é mostrado na figura 5.1.

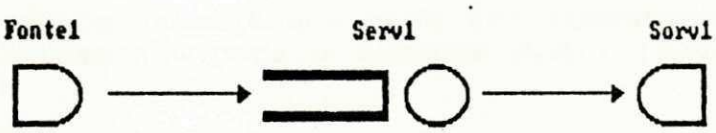


Figura 5.1 - Sistema M/M/1

- Definição dos elementos usando o SIM/SAVAD

Para modelar o sistema M/M/1 definem-se os seguintes elementos: uma fonte, uma estação de serviço, um sorvedouro e uma rota.

Os parâmetros adotados para estes elementos foram os seguintes:

Fonte:
- nome: Fonte1
- distribuição: exponencial
- média: 1.0

Estação de Serviço:
- nome: Serv1
- comprimento de fila: ilimitado
- servidor: único
- distribuição: exponencial
- média: 0.5

Sorvedouro:
- nome: Sorv1

Classe:
- nome: c1
- prioridade: 0

Rota:
- nome: rota1
- classe: c1
- sequência de nodos: Fonte1 >> Serv1 >> Sorv1

Devido a não repetição de elementos na rota, o SIM/SAVAD reconhece que a rota é aberta.

- Resumo dos Resultados

As tabelas 1 e 2 apresentam os resultados obtidos para o tempo médio de resposta (tempo médio de espera em fila adicionado ao tempo médio de serviço) e a utilização do servidor. Foram consideradas 4000 unidades de tempo simulado (uts), 10 execuções e nível de confiança igual a 90%. As sementes definidas para os geradores de números aleatórios foram as seguintes: 13, 31051, 2749, 24851, 6967, 31327, 10457, 14479, 29983, 7757.

Cada um destes resultados pode ser comparado com resultados obtidos analiticamente para o sistema M/M/1 [Moura 86], que são os seguintes:

- Utilização do servidor : 0.5

- Tempo médio de resposta: 1.0

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.90407	.....	.....	.....	.....
2	0.99045	0.9472622	0.06108	6.31400	28.78960
3	0.94471	0.9464102	0.04322	2.92000	7.69834
4	1.04294	0.9705420	0.05979	2.35300	7.24745
5	1.00569	0.9775713	0.05411	2.13200	5.27757
6	0.92620	0.9690087	0.05275	2.01500	4.47786
7	1.00758	0.9745190	0.05031	1.94300	3.79129
8	1.06170	0.9854161	0.05585	1.89500	3.79739
9	1.02072	0.9893385	0.05355	1.86000	3.35612
10	0.96559	0.9869637	0.05105	1.83300	2.99796

Tabela 1 - Tempo Médio De Resposta para o Sistema M/M/1

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.48372	.....	.....	.....	.....
2	0.51745	0.5005856	0.02385	6.31400	21.27140
3	0.48660	0.4959228	0.01870	2.92000	6.35645
4	0.52003	0.5019485	0.01945	2.35300	4.55897
5	0.49979	0.5015160	0.01687	2.13200	3.20772
6	0.48741	0.4991646	0.01615	2.01500	2.66201
7	0.51070	0.5008120	0.01538	1.94300	2.25477
8	0.49985	0.5006923	0.01424	1.89500	1.90544
9	0.50705	0.5013983	0.01349	1.86000	1.66777
10	0.48027	0.4992854	0.01436	1.83300	1.66766

Tabela 2 - Utilização do Servidor para o Sistema M/M/1

5.1.1.2 - Sistema M/M/2

A diferença entre este sistema e o sistema M/M/1 é que a estação de serviço Serv1 deve ser definida com 2 servidores. Esse sistema é mostrado na figura 5.2.

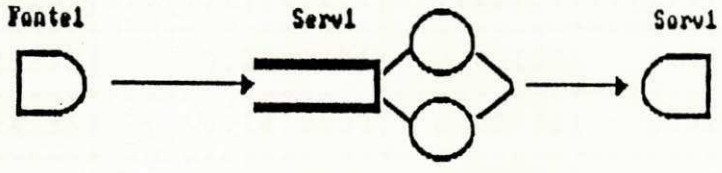


Figura 5.2 - Sistema M/M/2

As tabelas 3 e 4 apresentam os resultados obtidos para o tempo médio de resposta e a utilização dos servidores. Esses resultados foram obtidos considerando os mesmos parâmetros da simulação usados no sistema M/M/1, excetuando a estação de serviço, que nesse caso apresenta dois servidores, com o mesmo tempo médio de serviço.

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.50845	.....	.....	.....	.....
2	0.52860	0.5185264	0.01425	6.31400	12.26706
3	0.51543	0.5174934	0.01023	2.92000	3.33325
4	0.54009	0.5231427	0.01405	2.35300	3.16011
5	0.54444	0.5274022	0.01545	2.13200	2.79372
6	0.52076	0.5262950	0.01409	2.01500	2.20161
7	0.53603	0.5276857	0.01337	1.94300	1.86131
8	0.53915	0.5291193	0.01303	1.89500	1.64979
9	0.54441	0.5308185	0.01321	1.86000	1.54303
10	0.51757	0.5294932	0.01314	1.83300	1.43862

Tabela 3 - Tempo Médio De Resposta para o Sistema M/M/2

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.24187	.....	.....	.....	.....
2	0.25873	0.2502991	0.01192	6.31400	21.25502
3	0.24335	0.2479833	0.00933	2.92000	6.34410
4	0.26007	0.2510045	0.00972	2.35300	4.55806
5	0.24989	0.2507823	0.00844	2.13200	3.20746
6	0.24370	0.2496025	0.00808	2.01500	2.66300
7	0.25535	0.2504233	0.00769	1.94300	2.25492
8	0.24993	0.2503613	0.00712	1.89500	1.90562
9	0.25352	0.2507126	0.00674	1.86000	1.66774
10	0.24013	0.2496548	0.00718	1.83300	1.66809

Tabela 4 - Utilização dos Servidores para o Sistema M/M/2

Cada um destes resultados pode ser comparado com os resultados obtidos analiticamente para o sistema M/M/2 [Sauer 81:64], que são os seguintes:

- Utilização de cada servidor: 0.25

- Tempo médio de resposta : 0.5333

5.1.1.3 - Sistema M/M/1/5

A diferença entre este sistema e o sistema M/M/1 é que há um limite K para a quantidade de clientes que podem esperar na estação de serviço Serv1 (em fila e no serviço). Nesse caso o valor de K é 5.

Este sistema é mostrado na figura 5.3.

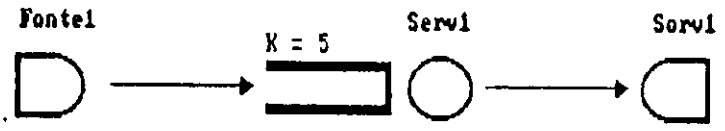


Figura 5.3 - Sistema M/M/1/5

As tabelas 5 e 6 apresentam os resultados obtidos para o tempo médio de resposta e a utilização dos servidores para o Sistema M/M/1/5. Esses resultados foram obtidos considerando os mesmos parâmetros de simulação utilizados para o sistema M/M/1, excetuando-se o comprimento de fila que foi considerado igual a 5.

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1	0.88043	.....	.....	.....	.....
2	0.92678	0.9036064	0.03277	6.31400	16.19300
3	0.87996	0.8957240	0.02690	2.92000	5.06229
4	0.95543	0.9106503	0.03706	2.35300	4.78794
5	0.94607	0.9177341	0.03579	2.13200	3.71844
6	0.88009	0.9114609	0.03551	2.01500	3.20484
7	0.94377	0.9160758	0.03464	1.94300	2.77688
8	0.92529	0.9172279	0.03223	1.89500	2.35455
9	0.93669	0.9193909	0.03084	1.86000	2.07992
10	0.85617	0.9130683	0.03529	1.83300	2.24028

Tabela 5 - Tempo Médio de Resposta para o Sistema M/M/1/5

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.47822	.....	.....	.....	.....
2	0.51033	0.4942760	0.02270	6.31400	20.50733
3	0.47846	0.4890045	0.01847	2.92000	6.36708
4	0.50667	0.4934196	0.01747	2.35300	4.16663
5	0.49408	0.4935524	0.01514	2.13200	2.92410
6	0.48240	0.4916944	0.01428	2.01500	2.38957
7	0.50243	0.4932278	0.01366	1.94300	2.03315
8	0.49238	0.4931215	0.01265	1.89500	1.71812
9	0.49836	0.4937034	0.01196	1.86000	1.50159
10	0.47548	0.4918814	0.01266	1.83300	1.49193

Tabela 6 - Utilização dos Servidores para o Sistema M/M/1/5

Cada um desses resultados pode ser comparado com os resultados obtidos analiticamente para o sistema M/M/1/5 [Kleinrock 75:104], que são os seguintes:

- Utilização do servidor : 0.4920633

- Tempo médio de resposta: 0.9047618

5.1.1.4 - Sistema M/M/oo

Nesse sistema sempre haverá um servidor disponível para um cliente que chega. Este sistema é mostrado na figura 5.4.

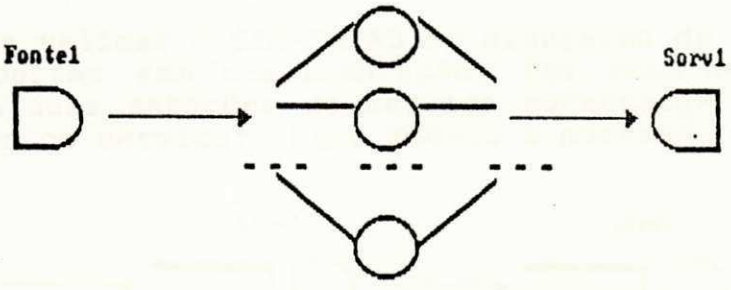


Figura 5.4 - Sistema M/M/oo

A tabela 7 apresenta os resultados obtidos para o número médio de servidores ocupados. Estes resultados foram obtidos considerando os mesmos parâmetros utilizados no sistema M/M/1, com exceção do número de servidores, que é ilimitado.

Esses resultados podem ser comparado com o resultado obtido analiticamente para o sistema M/M/oo [Kleinrock 76], que é o seguinte:

- Número médio de servidores ocupados: 0.5

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.48375	.....	.....	.....	.....
2	0.51745	0.5005983	0.02383	6.31400	21.25502
3	0.48670	0.4959667	0.01866	2.92000	6.34410
4	0.52014	0.5020091	0.01945	2.35300	4.55806
5	0.49979	0.5015646	0.01687	2.13200	3.20746
6	0.48741	0.4992050	0.01616	2.01500	2.66300
7	0.51070	0.5008467	0.01538	1.94300	2.25492
8	0.49985	0.5007226	0.01424	1.89500	1.90562
9	0.50705	0.5014253	0.01349	1.86000	1.66774
10	0.48027	0.4993096	0.01437	1.83300	1.66809

Tabela 7 - Número Médio de Servidores Ocupados no Sistema M/M/oo

5.1.2 - Modelos de Redes de Filas

5.1.2.1 - Rede Aberta sem Realimentação

Para validar o SIM/SAVAD na simulação de modelos de redes de filas abertas sem realimentação, foi considerado um modelo de rede com duas estações de serviço conectadas em série, cada uma com um único servidor. Esse modelo é mostrado na figura 5.5.

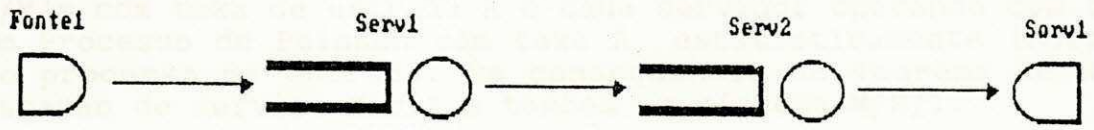


Figura 5.5 - Rede Aberta

- Definição dos elementos usando o SIM/SAVAD

Para modelar essa rede definem-se os seguintes elementos: uma fonte, duas estações de serviço, um sorvedouro e uma rota.

Os parâmetros adotados para estes elementos foram os seguintes:

Fonte:
- nome: Fonte1
- distribuição: exponencial
- média: 1.0

Primeira Estação de Serviço:
- nome: Serv1
- comprimento de fila: ilimitado
- servidor: único
- distribuição: exponencial
- média: 0.5

Segunda Estação de Serviço:
- nome: Serv2
- comprimento de fila: ilimitado
- servidor: único
- distribuição: exponencial
- média: 0.5

Sorvedouro:
- nome: Sorv1

Classe:
- nome: c1
- prioridade: 0

Rota:
- nome: rota1
- classe: c1
- sequência de nodos: Fonte1 >> Serv1 >> Serv2 >> Sorv1

- Resumo do Resultados

As tabelas 8 e 9 (10 e 11) apresentam os resultados obtidos respectivamente para o tempo médio de resposta e a utilização do servidor de Serv1 (Serv2). Foram consideradas 4000 uts, 11 execuções e nivel de confiança igual a 90%. As sementes definidas para os geradores de números aleatórios foram as seguintes: 13, 31, 31051, 5682, 2749, 13886, 24851, 61, 6967, 7993, 31327.

Conforme o Teorema de Burke [Moura 86] a saída de um sistema M/M/m com taxa de entrada Ä e cada servidor operando com taxa ß é um Processo de Poisson com taxa Ä, estatisticamente independente do processo de entrada. Em consequência do Teorema de Burke, a estação de serviço Serv2 é também um sistema M/M/1.

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.90407	.....	.....	.....	.....
2	0.80908	0.8565760	0.06717	6.31400	35.00912
3	0.99065	0.9012665	0.09082	2.92000	16.98746
4	0.94034	0.9110353	0.07668	2.35300	9.90252
5	0.94471	0.9177696	0.06809	2.13200	7.07417
6	1.05186	0.9401185	0.08189	2.01500	7.16567
7	1.04294	0.9548069	0.08425	1.94300	6.48038
8	1.05814	0.9677241	0.08614	1.89500	5.96349
9	1.00569	0.9719424	0.08156	1.86000	5.20276
10	1.00669	0.9754174	0.07768	1.83300	4.61603
11	0.92620	0.9709427	0.07517	1.81250	4.23097

Tabela 8 - Tempo Médio de Resposta de Serv1 para o Modelo Rede Aberta

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.48369	.....	.....	.....	.....
2	0.50000	0.4918468	0.01153	6.31400	10.46552
3	0.51763	0.5004426	0.01697	2.92000	5.71816
4	0.48116	0.4956211	0.01688	2.35300	4.00791
5	0.48660	0.4938162	0.01517	2.13200	2.92877
6	0.51775	0.4978055	0.01672	2.01500	2.76296
7	0.52003	0.5009798	0.01742	1.94300	2.55377
8	0.50276	0.5012020	0.01614	1.89500	2.15767
9	0.49979	0.5010448	0.01511	1.86000	1.86924
10	0.49714	0.5006539	0.01430	1.83300	1.65511
11	0.48741	0.4994496	0.01414	1.81250	1.54695

Tabela 9 - Utilização de Serv1 para o Modelo Rede Aberta

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.90407	.....	.....	.....	.....
2	0.80908	0.8565760	0.06717	6.31400	35.00912
3	0.99065	0.9012665	0.09082	2.92000	16.98746
4	0.94034	0.9110353	0.07668	2.35300	9.90252
5	0.94471	0.9177696	0.06809	2.13200	7.07417
6	1.05186	0.9401185	0.08189	2.01500	7.16567
7	1.04294	0.9548069	0.08425	1.94300	6.48038
8	1.05814	0.9677241	0.08614	1.89500	5.96349
9	1.00569	0.9719424	0.08156	1.86000	5.20276
10	1.00669	0.9754174	0.07768	1.83300	4.61603
11	0.92620	0.9709427	0.07517	1.81250	4.23097

Tabela 8 - Tempo Médio de Resposta de Serv1 para o Modelo Rede Aberta

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.48369	.....	.....	.....	.....
2	0.50000	0.4918468	0.01153	6.31400	10.46552
3	0.51763	0.5004426	0.01697	2.92000	5.71816
4	0.48116	0.4956211	0.01688	2.35300	4.00791
5	0.48660	0.4938162	0.01517	2.13200	2.92877
6	0.51775	0.4978055	0.01672	2.01500	2.76296
7	0.52003	0.5009798	0.01742	1.94300	2.55377
8	0.50276	0.5012020	0.01614	1.89500	2.15767
9	0.49979	0.5010448	0.01511	1.86000	1.86924
10	0.49714	0.5006539	0.01430	1.83300	1.65511
11	0.48741	0.4994496	0.01414	1.81250	1.54695

Tabela 9 - Utilização de Serv1 para o Modelo Rede Aberta

No processo de validação desse modelo, os resultados de Serv1 e Serv2 são comparados com os resultados obtidos analiticamente para o sistema M/M/1, que são os seguintes:

- Utilização do servidor de Serv1 (Serv2): 0.5

- Tempo médio de resposta de Serv1 (Serv2): 1.0

5.1.2.2 - Rede Fechada

Para validar o SIM/SAVAD na simulação de modelos de redes de filas fechadas escolheu-se um modelo de rede com duas estações de serviço conectadas em série, cada uma com um único servidor e com um número fixo de clientes (população) circulando na rede. Esse modelo é mostrado na figura 5.6.

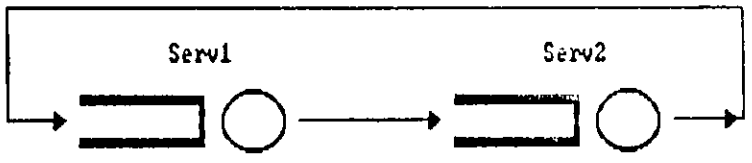


Figura 5.6 - Rede Fechada

- Definição de elementos no SIM/SAVAD

Para modelar essa rede definem-se os seguintes elementos: duas estações de serviço e uma rota.

Os parâmetros adotados para estes elementos foram os seguintes:

Primeira Estação de Serviço:
- nome: Serv1
- comprimento de fila: ilimitado
- servidor: único
- distribuição: exponencial
- média: 1.0

Segunda Estação de Serviço:
- nome: Serv2
- comprimento de fila: ilimitado
- servidor: único
- distribuição: exponencial
- média: 1.0

Classe:
- nome: cl
- prioridade: 0

Rota:
- nome: rota1
- classe: c
- sequência de nodos: Serv1 >> Serv2 >> Serv1

Por ser uma rota fechada, o primeiro nó foi repetido na rota. Nesse caso, a Interface solicita a população da rede, tendo sido atribuído o valor 2 (dois).

- Resumo dos Resultados

As tabelas 12 e 13 (14 e 15) apresentam os resultados obtidos respectivamente para o tempo médio de resposta e a utilização do servidor de Serv1 (Serv2). Foram consideradas 4000 uts, 10 execuções e nível de confiança igual a 90%. As sementes definidas para os geradores de amostras foram as seguintes: 13, 31, 31051, 5682, 2749, 13886, 24851, 61, 6967, 7993.

Cada um destes resultados foi comparado com resultados obtidos analiticamente, através do método de Análise do Valor Médio - AVM [Moura 86:272]. Os resultados obtidos pelo AVM são os seguintes:

- Utilização do Servidor de Serv1 (Serv2): 0.666

- Tempo Médio de Resposta de Serv1 (Serv2): 1.5

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	1.41981	.....	.....	.....	.....
2	1.55479	1.4872979	0.09545	6.31400	28.65148
3	1.45853	1.4777102	0.06950	2.92000	7.92935
4	1.48414	1.4793186	0.05684	2.35300	4.52049
5	1.43522	1.4704994	0.05303	2.13200	3.43831
6	1.51952	1.4786692	0.05148	2.01500	2.86390
7	1.51413	1.4837344	0.04887	1.94300	2.41871
8	1.51149	1.4872040	0.04629	1.89500	2.08555
9	1.50500	1.4891815	0.04371	1.86000	1.81975
10	1.50971	1.4912341	0.04172	1.83300	1.62155

Tabela 12 - Tempo Médio De Resposta de Serv1 para o Modelo Rede Fechada

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.63688	.....	.....	.....	.....
2	0.68607	0.6614753	0.03478	6.31400	23.47511
3	0.65424	0.6590634	0.02495	2.92000	6.38100
4	0.66921	0.6616001	0.02099	2.35300	3.73263
5	0.66148	0.6615763	0.01818	2.13200	2.61983
6	0.68027	0.6646920	0.01796	2.01500	2.22286
7	0.66904	0.6653136	0.01648	1.94300	1.81892
8	0.66555	0.6653427	0.01526	1.89500	1.53627
9	0.67119	0.6659920	0.01440	1.86000	1.34086
10	0.67177	0.6665699	0.01370	1.83300	1.19151

Tabela 13 - Utilização de Serv1 para o Modelo Rede Fechada

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	1.56402	.....	.....	.....	.....
2	1.41778	1.4909014	0.10341	6.31400	30.96777
3	1.53886	1.5068861	0.07819	2.92000	8.74758
4	1.47869	1.4998375	0.06538	2.35300	5.12845
5	1.48606	1.4970820	0.05695	2.13200	3.62729
6	1.43917	1.4874307	0.05616	2.01500	3.10590
7	1.50665	1.4901762	0.05178	1.94300	2.55174
8	1.51809	1.4936650	0.04894	1.89500	2.19533
9	1.50238	1.4946334	0.04587	1.86000	1.90293
10	1.50376	1.4955456	0.04335	1.83300	1.68002

Tabela 14 - Tempo Médio De Resposta de Serv2 para o Modelo Rede Fechada

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.68477	.....	.....	.....	.....
2	0.63979	0.6622823	0.03181	6.31400	21.44324
3	0.68094	0.6685013	0.02494	2.92000	6.28908
4	0.66768	0.6682953	0.02037	2.35300	3.58537
5	0.67905	0.6704469	0.01828	2.13200	2.59993
6	0.65274	0.6674952	0.01788	2.01500	2.20341
7	0.66645	0.6673463	0.01633	1.94300	1.79661
8	0.66743	0.6673562	0.01511	1.89500	1.51745
9	0.67018	0.6676700	0.01417	1.86000	1.31584
10	0.66922	0.6678253	0.01337	1.83300	1.16035

Tabela 15 - Utilização de Serv2 para o Modelo Rede Fechada

5.1.2.3 - Rede Aberta com Ponto de Duplicação

Para validar o uso do elemento ponto de duplicação escolheu-se um modelo de rede com uma fonte, três estações de serviço, cada uma com um servidor, um ponto de duplicação e dois sorvedouros. Esse modelo é mostrado na figura 5.7.

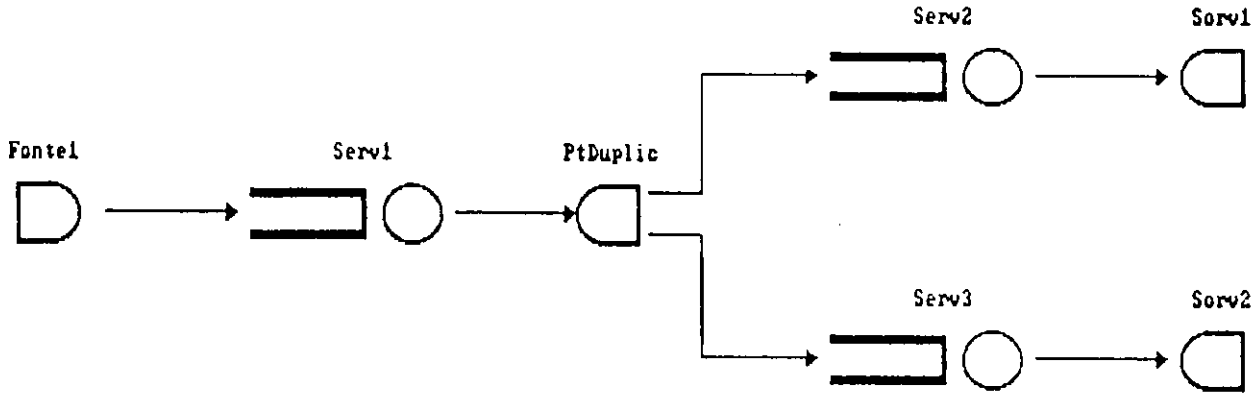


Figura 5.7 - Rede Aberta com Ponto de Duplicacao

- Definição de elementos no SIM/SAVAD

Os parâmetros adotados na definição dos elementos de modelagem do SIM/SAVAD foram os seguintes:

Fonte:
- nome: Fonte1
- distribuição: exponencial
- média: 1.0

Primeira Estação de Serviço:
- nome: Serv1
- comprimento de fila: ilimitado
- servidor: único
- distribuição: exponencial
- média: 0.5

Segunda Estação de Serviço:
- nome: Serv2
- comprimento de fila: ilimitado
- servidor: único
- distribuição: exponencial
- média: 0.5

Terceira Estação de Serviço:
- nome: Serv3
- comprimento de fila: ilimitado
- servidor: único
- distribuição: exponencial
- média: 0.5

Primeiro Sorvedouro:
- nome: Sorv1

Segundo Sorvedouro:
- nome: Sorv2

Ponto de Duplicação:
- nome: PtDuplic

Classe:
- nome: c1
- prioridade: 0

Primeira Rota:
- nome: rota1
- classe: c1
- sequência de nodos:
  Fonte1 >> Serv1 >> PtDuplic >> Serv2 >> Sorv1

Segunda Rota:
- nome: rota2
- classe: c1
- sequência de nodos: PtDuplic >> Serv3 >> Sorv2

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.97451	.....	.....	.....	.....
2	1.11684	1.0456753	0.10064	6.31400	42.96952
3	1.04328	1.0448765	0.07118	2.92000	11.48397
4	0.98160	1.0290562	0.06617	2.35300	7.56510
5	0.92077	1.0073996	0.07503	2.13200	7.10088
6	0.97600	1.0021670	0.06832	2.01500	5.60786
7	1.06967	1.0118107	0.06738	1.94300	4.89076
8	1.05051	1.0166482	0.06387	1.89500	4.20895
9	0.99268	1.0139846	0.06027	1.86000	3.68549
10	1.05246	1.0178320	0.05812	1.83300	3.30961
11	1.01315	1.0174062	0.05515	1.81250	2.96239

Tabela 10 - Tempo Médio de Resposta de Serv2 para o Modelo Rede Aberta

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.50658	.....	.....	.....	.....
2	0.47775	0.4921657	0.02038	6.31400	18.48778
3	0.52661	0.5036457	0.02456	2.92000	8.22002
4	0.49405	0.5012476	0.02062	2.35300	4.83896
5	0.48700	0.4983982	0.01896	2.13200	3.62659
6	0.50037	0.4987270	0.01697	2.01500	2.79991
7	0.51713	0.5013562	0.01699	1.94300	2.48804
8	0.50180	0.5014112	0.01573	1.89500	2.10135
9	0.50110	0.5013772	0.01471	1.86000	1.81916
10	0.49849	0.5010885	0.01390	1.83300	1.60788
11	0.49751	0.5007634	0.01323	1.81250	1.44385

Tabela 11 - Utilização de Serv2 para o Modelo Rede Aberta

Nesse modelo, os clientes da rota1 que saem da estação de serviço Serv1 são duplicados em PtDuplic; desta forma surgem os clientes da rota2.

Da mesma forma que no modelo de Redes de Filas Abertas sem Realimentação, pode-se usar o Teorema de Burke e afirmar que o Processo de saída da estação de serviço Serv1 é também um Processo de Poisson.

Como o fluxo de clientes entre Serv1 e Serv2 não têm suas características alteradas pelo ponto de duplicação, pelo Teorema de Burke, a estação de serviço Serv2 é também um sistema M/M/1.

Como o processo de chegada em Serv3 é idêntico ao processo de chegada em Serv2, Serv3 é também um sistema M/M/1.

No processo de validação desse modelo, os resultados de Serv1, Serv2 e Serv3 são comparados com os resultados obtidos analiticamente para o sistema M/M/1, que são os seguintes:

- Utilização do servidor de Serv1 (Serv2 e Serv3): 0.5

- Tempo médio de resposta de Serv1 (Serv2 e Serv3): 1.0

- Resumo dos Resultados

As tabelas 16 e 17 (18 e 19) (20 e 21) apresentam os resultados obtidos respectivamente para o tempo médio de resposta e a utilização do servidor de Serv1 (Serv2) (Serv3).Foram consideradas 4000 uts, 10 execuções e nível de confiança igual a 90%. As sementes definidas para os geradores de números aleatórios foram as seguintes: 19471, 23297, 23087, 31549, 18973, 31051, 2749, 24851, 6967, 31327.

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	1.02044	.....	.....	.....	.....
2	0.97820	0.9993179	0.02987	6.31400	13.34545
3	0.98136	0.9933310	0.02353	2.92000	3.99348
4	0.98655	0.9916345	0.01951	2.35300	2.31466
5	0.98829	0.9909655	0.01696	2.13200	1.63198
6	0.95442	0.9848744	0.02128	2.01500	1.77729
7	0.91604	0.9750410	0.03247	1.94300	2.44544
8	1.03021	0.9819378	0.03583	1.89500	2.44500
9	1.00674	0.9846941	0.03452	1.86000	2.17381
10	0.99158	0.9853832	0.03262	1.83300	1.91903

Tabela 16 - Tempo Médio de Resposta de Serv1 para o Modelo Rede Aberta com Ponto De Duplicação

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.49520	.....	.....	.....	.....
2	0.48605	0.4906289	0.00647	6.31400	5.88823
3	0.51267	0.4979752	0.01352	2.92000	4.57771
4	0.50559	0.4998778	0.01168	2.35300	2.74846
5	0.50418	0.5007387	0.01029	2.13200	1.96026
6	0.49552	0.4998692	0.00945	2.01500	1.55535
7	0.47735	0.4966527	0.01212	1.94300	1.79193
8	0.51303	0.4986995	0.01263	1.89500	1.69614
9	0.50784	0.4997152	0.01220	1.86000	1.51323
10	0.50752	0.5004961	0.01176	1.83300	1.36211

Tabela 17 - Utilização de Serv1 para o Modelo Rede Aberta com Ponto De Duplicação

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.90811	.....	.....	.....	.....
2	0.99858	0.9533467	0.06397	6.31400	29.95864
3	0.76850	0.8917307	0.11591	2.92000	21.91388
4	1.00054	0.9189319	0.10916	2.35300	13.97616
5	1.06612	0.9483702	0.11520	2.13200	11.58164
6	0.96554	0.9512312	0.10327	2.01500	8.93113
7	0.97913	0.9552174	0.09486	1.94300	7.29331
8	0.92815	0.9518340	0.08835	1.89500	6.21864
9	0.99124	0.9562120	0.08368	1.86000	5.42564
10	1.02051	0.9626422	0.08147	1.83300	4.90570

Tabela 18 - Tempo Médio de Resposta de Serv2 para o Modelo Rede Aberta com Ponto De Duplicação

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.47893	.....	.....	.....	.....
2	0.49950	0.4892142	0.01454	6.31400	13.27200
3	0.49987	0.4927654	0.01198	2.92000	4.09945
4	0.49957	0.4944654	0.01036	2.35300	2.46440
5	0.51863	0.4992982	0.01404	2.13200	2.68189
6	0.49032	0.4978013	0.01309	2.01500	2.16243
7	0.49052	0.4967611	0.01226	1.94300	1.81224
8	0.49185	0.4961470	0.01148	1.89500	1.55041
9	0.50719	0.4973737	0.01135	1.86000	1.41518
10	0.52043	0.4996796	0.01295	1.83300	1.50239

Tabela 19 - Utilização de Serv2 para o Modelo de Rede Aberta com Ponto De Duplicação

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.97451	.....	.....	.....	.....
2	1.11684	1.0456753	0.10064	6.31400	42.96952
3	1.04328	1.0448765	0.07118	2.92000	11.48397
4	0.98160	1.0290562	0.06617	2.35300	7.56510
5	0.92077	1.0073996	0.07503	2.13200	7.10088
6	0.97600	1.0021670	0.06832	2.01500	5.60786
7	1.06967	1.0118107	0.06738	1.94300	4.89076
8	1.05051	1.0166482	0.06387	1.89500	4.20895
9	0.99268	1.0139846	0.06027	1.86000	3.68549
10	1.05246	1.0178320	0.05812	1.83300	3.30961

Tabela 20 - Tempo Médio de Resposta de Serv3 para o Modelo Rede Aberta com Ponto De Duplicação

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.49378	.....	.....	.....	.....
2	0.48553	0.4896558	0.00584	6.31400	5.32121
3	0.51435	0.4978879	0.01484	2.92000	5.02609
4	0.48555	0.4948041	0.01360	2.35300	3.23339
5	0.51229	0.4983005	0.01414	2.13200	2.70475
6	0.50527	0.4994622	0.01296	2.01500	2.13447
7	0.48518	0.4974223	0.01300	1.94300	1.91981
8	0.50427	0.4982783	0.01228	1.89500	1.65116
9	0.48641	0.4969600	0.01215	1.86000	1.51566
10	0.51995	0.4992593	0.01357	1.83300	1.57515

Tabela 21 - Utilização de Serv3 para o Modelo de Rede Aberta com Ponto De Duplicação

5.1.2.4 - Rede Aberta com Ponto de Fusão

Para validar o uso do elemento ponto de fusão escolheu-se um modelo de rede com duas fontes, três estações de serviço, onde duas são do tipo servidor único e uma do tipo servidor múltiplo (com dez servidores), um ponto de fusão e um sorvedouro. Esse modelo é mostrado na figura 5.8.

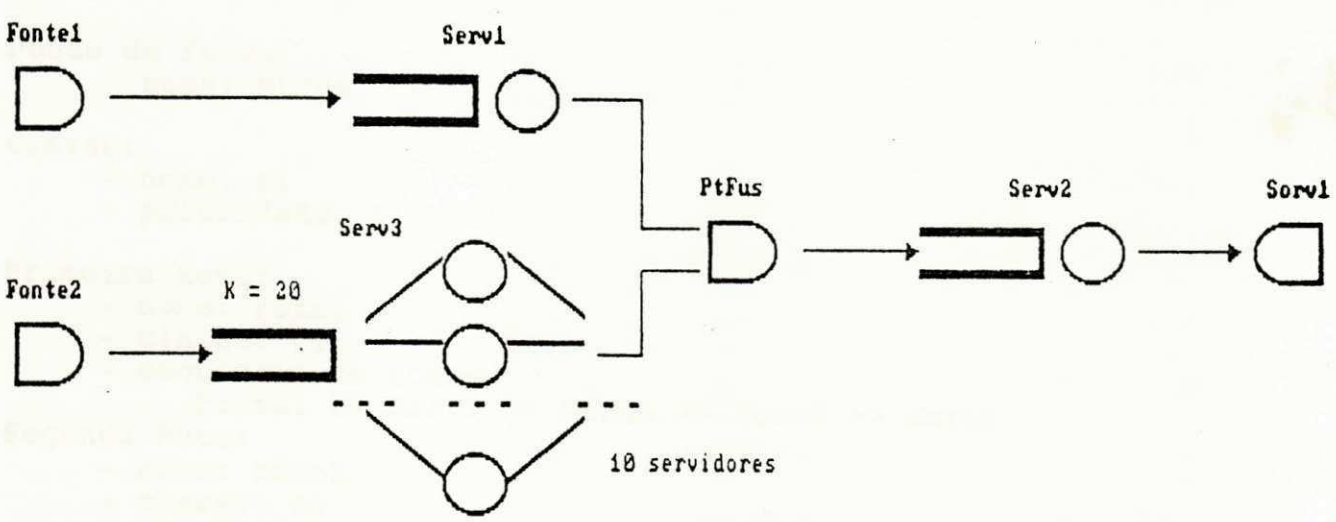


Figura 5.8 - Rede Aberta com Ponto de Fusao

- Definição de elementos no SIM/SAVAD

Os parâmetros adotados na definição dos elementos de modelagem do SIM/SAVAD foram os seguintes:

Primeira Fonte:
- nome: Fonte1
- distribuição: exponencial
- média: 1.0

Segunda Fonte:
- nome: Fonte2
- distribuição: determinística
- valor 0.8

Primeira Estação de Serviço:
- nome: Serv1
- comprimento de fila: ilimitado
- servidor: único
- distribuição: exponencial
- média: 0.5

Segunda Estação de Serviço:
- nome: Serv2
- comprimento de fila: ilimitado
- servidor: único
- distribuição: exponencial
- média: 0.5

Terceira Estação de Serviço:
- nome: Serv3
- comprimento de fila: 20
- servidor: multiplo
- número de servidores: 10
- distribuição: determinística
- média: 0.01

Sorvedouro:
- nome: Sorv1

Ponto de Fusão:
- nome: PtFus

Classe:
- nome: c1
- prioridade: 0

Primeira Rota:
- nome: rota1
- classe: c1
- sequência de nodos:
  Fonte1 >> Serv1 >> PtFus >> Serv2 >> Sorv1

Segunda Rota:
- nome: rota2
- classe: c1
- sequência de nodos: Fonte2 >> Serv3 >> PtFus

Nesse modelo é preciso que um cliente tenha terminado serviço em Serv1 e outro cliente tenha terminado serviço em Serv3 para que o elemento PtFus proceda uma fusão. Pode-se observar que os parâmetros adotados em Fonte2 e Serv3 fazem com que PtFus sempre tenha um cliente em Serv3 esperando a chegada de um cliente de Serv1. Dessa forma o fluxo de saída de clientes de PtFus é determinado pelo fluxo de saída de Serv1.

Nesse modelo a rota rota1 corresponde a uma cadeia aberta sem realimentação, podendo-se aplicar o Teorema de Burke. Os resultados das medidas de desempenho dos elementos Serv1 e Serv2 podem ser comparados com os resultados obtidos analiticamente para o sistema M/M/1, que são os seguintes:

- Utilização do servidor de Serv1 (Serv2): 0.5

- Tempo médio de resposta de Serv1 (Serv2): 1.0

- Resumo dos Resultados

As tabelas 22 e 23 (24 e 25) apresentam os resultados obtidos para o tempo médio de resposta e a utilização do servidor de Serv1 (Serv2).

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.90508	.....	.....	.....	.....
2	0.80994	0.8575099	0.06727	6.31400	35.02548
3	0.99065	0.9018891	0.09040	2.92000	16.89719
4	0.94034	0.9115022	0.07627	2.35300	9.84443
5	0.94575	0.9183524	0.06780	2.13200	7.03970
6	1.05212	0.9406466	0.08161	2.01500	7.13702
7	1.04271	0.9552272	0.08389	1.94300	6.44987
8	1.05893	0.9681900	0.08589	1.89500	5.94356
9	1.00610	0.9724019	0.08133	1.86000	5.18560
10	1.00829	0.9759903	0.07751	1.83300	4.60360
11	0.93050	0.9718551	0.07480	1.81250	4.20637

Tabela 22 - Tempo Médio de Resposta de Serv1 para o Modelo Rede Aberta com Ponto De Fusão

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.48427	.....	.....	.....	.....
2	0.50081	0.4925390	0.01170	6.31400	10.60502
3	0.51763	0.5009040	0.01668	2.92000	5.61526
4	0.48116	0.4959671	0.01682	2.35300	3.99099
5	0.48726	0.4942251	0.01508	2.13200	2.90965
6	0.51778	0.4981513	0.01657	2.01500	2.73579
7	0.51994	0.5012638	0.01722	1.94300	2.52287
8	0.50347	0.5015392	0.01596	1.89500	2.13227
9	0.49998	0.5013662	0.01494	1.86000	1.84750
10	0.49824	0.5010532	0.01412	1.83300	1.63350
11	0.48962	0.5000142	0.01383	1.81250	1.51174

Tabela 23 - Utilização de Serv1 para o Modelo Rede Aberta com Ponto De Fusão

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.97385	.....	.....	.....	.....
2	1.11642	1.0451308	0.10081	6.31400	43.06594
3	1.04328	1.0445135	0.07129	2.92000	11.50687
4	0.98160	1.0287839	0.06617	2.35300	7.56682
5	0.92071	1.0071696	0.07496	2.13200	7.09658
6	0.97600	1.0019754	0.06825	2.01500	5.60297
7	1.06993	1.0116835	0.06739	1.94300	4.89163
8	1.05022	1.0165004	0.06386	1.89500	4.20895
9	0.99268	1.0138532	0.06026	1.86000	3.68505
10	1.05203	1.0176712	0.05808	1.83300	3.30824
11	1.01212	1.0171666	0.05513	1.81250	2.96178

Tabela 24 - Tempo Médio de Resposta de Serv2 para o Modelo Rede Aberta com Ponto De Fusão

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.50650	.....	.....	.....	.....
2	0.47775	0.4921259	0.02032	6.31400	18.43826
3	0.52661	0.5036193	0.02455	2.92000	8.21888
4	0.49405	0.5012277	0.02061	2.35300	4.83756
5	0.48700	0.4983823	0.01895	2.13200	3.62507
6	0.50040	0.4987186	0.01697	2.01500	2.79884
7	0.51713	0.5013491	0.01698	1.94300	2.48746
8	0.50180	0.5014051	0.01572	1.89500	2.10086
9	0.50115	0.5013764	0.01471	1.86000	1.81870
10	0.49848	0.5010868	0.01390	1.83300	1.60750
11	0.49766	0.5007756	0.01322	1.81250	1.44308

Tabela 25 - Utilização de Serv2 para o Modelo Rede Aberta com Ponto De Fusão

Foram consideradas 4000 uts, 11 execuções e nível de confiança igual a 90%. As sementes definidas para os geradores de números aleatórios foram as seguintes: 13, 31, 31051, 5682, 2749, 13886, 24851, 61, 6967, 7993, 31327.

5.1.2.5 - Rede de Filas com Ponto de Sincronização

Para validar o SIM/SAVAD na simulação de modelos que utilizam o elemento Ponto de Sincronização, foram escolhidos dois modelos apresentados em [Cabral 87] para o Protocolo da Camada de Sessão. Estes modelos foram solucionados através de Cadeias de Markov.

- Protocolo da Camada de Sessão

A seguir são analisadas duas configurações de serviço definidas para o protocolo de sessão do RM-OSI [Moura 86]: uma é a configuração mínima de serviços de sessão com o serviço de transferência normal de dados [Cabral 87:48], e a outra é uma configuração com transferência normal de dados e serviço quarentena de dados [Cabral 87:63].

A escolha de uma configuração de serviços de sessão depende da aplicação considerada. Têm-se como exemplos as aplicações

transferência de arquivos e o serviço telecompras, respectivamente direcionados à primeira e à segunda configurações de serviço mencionadas.

5.1.2.5.1 - Modelo do Protocolo de Sessão com Diálogo Semi-Duplex.

Considera-se nesse modelo uma conexão de sessão com diálogo semi-duplex, com taxa média de chegada de 12 (doze) Unidades de Dados do Serviço de Sessão (UDSSs) por segundo na entidade de sessão de referência. A capacidade de armazenamento desta entidade é de 6 (seis) Unidades de Dados do Protocolo de Sessão (UDPSs). Adota-se a disciplina de liberação de UDPSs do tipo limitada permitindo liberar até 2 (duas) UDPSs. A taxa média de transmissão na conexão de transporte associada a essa conexão de sessão é de 10 (dez) Unidades de Dados do Serviço de Transporte (UDSTs) por segundo. A conexão de transporte tem capacidade de armazenamento igual a 6 (seis) Unidades de Dados do Protocolo de Transporte (UDPTs). Na entidade de sessão par tem-se taxa média de chegada de 20 (vinte) UDSSs por segundo. A distribuição de probabilidade considerada é a exponencial.

A entidade de sessão de referência somente envia uma UDPS à entidade de sessão par quando recebe a ficha de dados.

A entidade de sessão de referência transferindo UDPSs libera a ficha de dados juntamente com a última UDPSs emitida. A entidade de sessão par, a medida que recebe UDPSs entrega-as (como UDSSs) ao usuário SS receptor. A entrega da última UDSS é seguida da geração de uma UDSS de resposta (e/ou reconhecimento). Essa UDSS juntamente com a ficha de dados são enviadas pela conexão de sessão à entidade de sessão de referência.

Esse modelo é mostrado na figura 5.9.

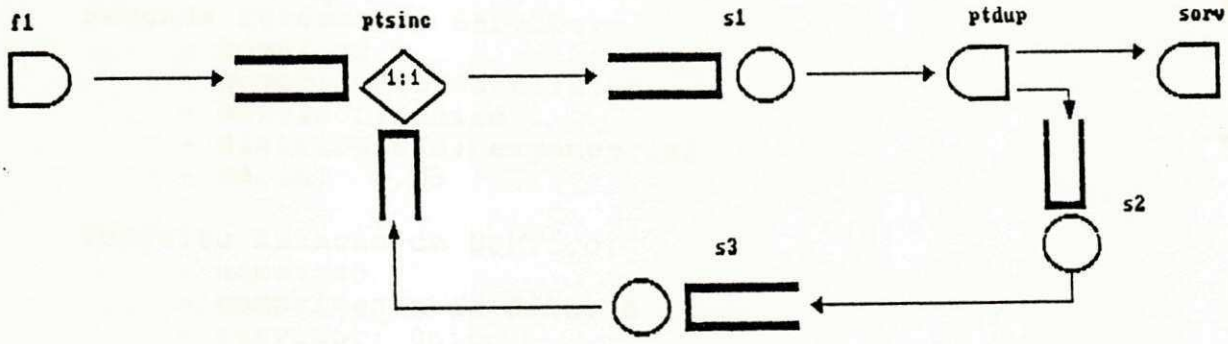


Figura 5.9 - Modelo do Protocolo de Sessao com Dialogo Semi-Duplex

A entidade de sessão de referência é modelada pelos seguintes elementos:

- uma fonte (f1) que representa a geração de UDSSs,
- um ponto de sincronização (ptsinc) que representa o processo de liberação de UDPSs com a ficha de dados para a entidade de sessão par.

A entidade de sessão par é modelada pelos seguintes elementos:

- um ponto de duplicação (ptdup),
- um sorvedouro (sorv),
- uma estação de serviço (s2).

Finalmente, a conexão de transporte, associada à conexão de sessão, é modelada por duas estações de serviço (s1 e s3).

- Definição de Elementos no SIM/SAVAD

A seguir, apresentam-se os elementos do modelo considerando uma unidade de tempo simulado (uts) igual a 1 segundo.

Fonte:
- nome: f1
- distribuição: exponencial
- média: 0,08333

Primeira Estação de Serviço:
- nome: s1
- comprimento de fila: 6
- servidor: único
- distribuição: exponencial
- média: 0,1

Segunda Estação de Serviço:
- nome: s2
- comprimento de fila: 6
- servidor: único
- distribuição: exponencial
- média: 0,05

Terceira Estação de Serviço:
- nome: s3
- comprimento de fila: 6
- servidor: único
- distribuição: exponencial
- média: 0,1

Ponto de Sincronização:
- nome: ptsinc
- número de filas: 2
- comprimento das filas: 6
- associação: 1:1

Ponto de Duplicação:
- nome ptdup

Sorvedouro:
- nome: sorv

Classe:
- nome: cl
- prioridade: 0

Primeira Rota:
- nome: rota1
- classe: cl
- sequência de nodos: f1 >> ptsinc.1

Segunda Rota:
- nome: rota2
- classe: cl
- sequência de nodos: ptdup >> sorv

Terceira Rota:
- nome: rota3
- classe: cl
- sequência de nodos:
  ptsinc.2 >> s1 >> ptdup >> s2 >> s3 >> ptsinc.2
- população: 2

- Resumo dos Resultados

As tabelas 26 e 27 apresentam os resultados obtidos para o atraso médio de admissão (tempo médio de espera na fila 1 do ponto de sincronização) e o tempo médio de transmissão de UDPSs na conexão de sessão (tempo médio de resposta na estação de serviço s1).

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.89347	.....	.....	.....	.....
2	0.88953	0.8915004	0.00279	6.31400	1.39692
3	0.86284	0.8819458	0.01667	2.92000	3.18579
4	0.86096	0.8766999	0.01718	2.35300	2.30587
5	0.88317	0.8779937	0.01516	2.13200	1.64623
6	0.87664	0.8777685	0.01357	2.01500	1.27176
7	0.87864	0.8778930	0.01239	1.94300	1.03664

Tabela 26 - Tempo Médio de Espera na Fila 1 de ptsinc no Modelo Protocolo de Sessão com Diálogo Semi-Duplex

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.14112	.....	.....	.....	.....
2	0.13825	0.1396895	0.00203	6.31400	6.48531
3	0.13474	0.1380381	0.00320	2.92000	3.90826
4	0.14048	0.1386486	0.00288	2.35300	2.44731
5	0.14228	0.1393743	0.00298	2.13200	2.03762
6	0.13297	0.1383077	0.00373	2.01500	2.21939
7	0.14623	0.1394395	0.00454	1.94300	2.38869

Tabela 27 - Tempo Médio de Resposta de s1 no Modelo Protocolo de Sessão com Diálogo Semi-Duplex

Foram consideradas 400 uts, 7 execuções e nível de confiança igual a 90%. As sementes definidas para os geradores de números aleatórios foram as seguintes: 13, 31051, 2749, 24851, 6967, 31327, 10457.

Cada um destes resultados pode ser comparado com resultados obtidos em [Cabral 87], que são os seguintes:

- Atraso Médio de Admissão : 0.88

- Tempo Médio de Transmissão: 0.14

5.1.2.5.2 - Modelo do Protocolo de Sessão com Quarentena de Dados Local e Modo de Diálogo Duplex

Nesse modelo, UDSSs chegam à entidade de sessão de referência e são quarentenadas (como UDPSs) até a composição de uma Unidade de Quarentena de Dados (UQD), quando então podem ser liberadas e enviadas pela conexão de sessão à entidade de sessão par. UDPSs que chegam à entidade de sessão par são entregues ao usuário SS receptor. Uma UDPS corresponde a uma UDSS.

A capacidade de armazenamento na entidade de sessão de referência é de 6 UDPSs. A taxa média de transmissão na conexão de transporte atribuída a esta conexão de sessão é de 10 UDSTs por segundo. A conexão de transporte tem capacidade de armazenamento de 6 UDSTs.

A figura 5.10 representa esse modelo. Nessa figura uma fonte (f1) representa a geração de UDSSs, um ponto de sincronização (ptsinc) representa o processo de formação e liberação de uma

UQD, uma estação de serviço (s1) representa a conexão de transporte, e um sorvedouro (sorv) representa a entrega de UDSSs à entidade de sessão par.

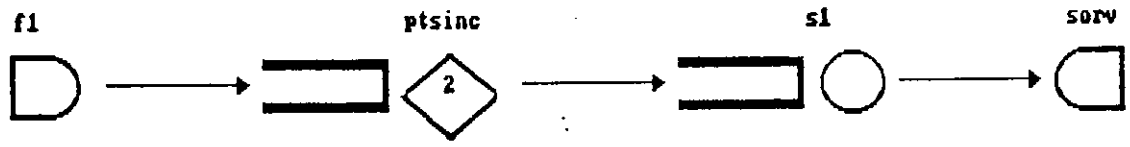


Figura 5.10 - Modelo do Protocolo de Sessao com Quarentena de Dados e Modo de Dialogo Duplex

- Definição de Elementos no SIM/SAVAD

A seguir, apresentam-se os elementos desse modelo considerando uma unidade de tempo simulado (uts) igual a 1 segundo.

Fonte:
- nome: f1
- distribuição: exponencial
- média: 0,08333

Estação de Serviço:
- nome: s1
- comprimento de fila: 6
- servidor: único
- distribuição: exponencial
- média: 0,1

Ponto de Sincronização:
- nome: ptsinc
- número de filas: 1
- comprimento da fila: 6
- associação: 2

Sorvedouro:
- nome: sorv

Classe:
- nome: c1
- prioridade: 0

Rota:
- nome: rota1
- classe: c1
- sequência de nodos: f1 >> ptsinc.1 >> s1 >> sorv

- Resumo dos Resultados

As tabelas 28 e 29 apresentam os resultados obtidos para o atraso médio de admissão mais atraso médio de quarentena de dados

(tempo médio de espera na fila 1 do ponto de sincronização) e o atraso médio fim-a-fim de uma UDSS (tempo médio de permanência dos clientes da rota r1 no modelo). Foram consideradas 400 uts, 7 execuções e nível de confiança igual a 90%. As sementes definidas para os geradores de números aleatórios foram as seguintes: 13, 31051, 2749, 24851, 6967, 31327, 10457.

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.31710	.....	.....	.....	.....
2	0.30888	0.3129921	0.00581	6.31400	8.29318
3	0.28848	0.3048218	0.01474	2.92000	8.15012
4	0.33205	0.3116278	0.01817	2.35300	6.85882
5	0.31173	0.3116475	0.01573	2.13200	4.81354
6	0.29509	0.3088883	0.01561	2.01500	4.15755
7	0.31952	0.3104071	0.01481	1.94300	3.50309

Tabela 28 - Tempo Médio de Espera na Fila 1 de ptsinc no Modelo Protocolo de Sessão com Quarentena De Dados

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.81711	.....	.....	.....	.....
2	0.81420	0.8156580	0.00206	6.31400	1.12768
3	0.76848	0.7999311	0.02728	2.92000	5.74901
4	0.84902	0.8122041	0.03314	2.35300	4.80114
5	0.81897	0.8135577	0.02886	2.13200	3.38270
6	0.78746	0.8092074	0.02793	2.01500	2.83920
7	0.83271	0.8125649	0.02700	1.94300	2.44012

Tabela 29 - Tempo Médio de Resposta de s1 no Modelo Protocolo de Sessão com Quarentena De Dados

Cada um destes resultados pode ser comparado com resultados obtidos em [Cabral 87], que são os seguintes:

- Tempo Médio de Espera na fila1 de ptsinc (Atraso Médio de Admissão mais Atraso Médio de Quarentena de Dados): 0.3

- Atraso Médio Fim-a-Fim: 0.75

5.2 - Comparação com Outros Simuladores

Até aqui, os modelos simulados pelo SIM/SAVAD foram solucionados através de técnicas analíticas. Os modelos que serão apresentados a seguir apresentam um maior grau de complexidade, dificultando a utilização de técnicas analíticas na busca de suas soluções. Dessa forma, para validar esses modelos, os seus resultados são comparados com aqueles obtidos através da execução em outros simuladores.

5.2.1 - Modelos usando Ponto Escalonador

Como exemplos foram escolhidos dois modelos de Protocolos de Acesso ao Meio apresentados em [Moura 86]: Protocolo de Passagem de Ficha numa Topologia em Anel e Protocolo CSMA/CD numa Topologia em Barramento.

- Sub-rede de Comunicação em Estudo.

Nos dois exemplos mostrados a seguir, a sub-rede possui uma topologia (anel e barramento), um protocolo de acesso ao meio de transmissão e N interfaces conectadas a sub-rede. Os usuários da rede local trocam pacotes de dados.

- Protocolo de Passagem de Ficha numa Topologia em Anel

Nesse exemplo, segundo [Moura 86:304], "... uma ficha (representada por uma sequência de bits particular) é passada ao redor do anel, de uma interface para outra. Qualquer interface, ao receber a ficha, pode removê-la do anel, transmitir um pacote, e então passar a ficha para a próxima interface."

- Protocolo CSMA/CD numa Topologia em Barramento

Nesse exemplo, segundo [Moura 86:324], "... cada interface monitora o meio de transmissão e transmite apenas quando o meio está desocupado. As interfaces monitoram as próprias transmissões e cessam de transmitir no instante em que detectam a presença de outras transmissões no meio. Neste caso, diz-se que ocorreu uma colisão. A fim de se resolverem colisões, as tentativas de retransmissão são marcadas para o futuro de acordo com uma função de distribuição de atraso chamada ´função de retirada´."

5.2.1.1 - Modelo do Protocolo de Passagem de Ficha

- Modelo de Moura.

O modelo apresentado em [Moura 86:304], para uma sub-rede em anel com protocolo de acesso ao meio com mecanismo de passagem de ficha é mostrado na figura 5.11. Neste modelo, clientes correspondem aos pacotes que chegam a sub-rede, filas modelam as interfaces e uma estação de serviço modela o meio de transmissão.

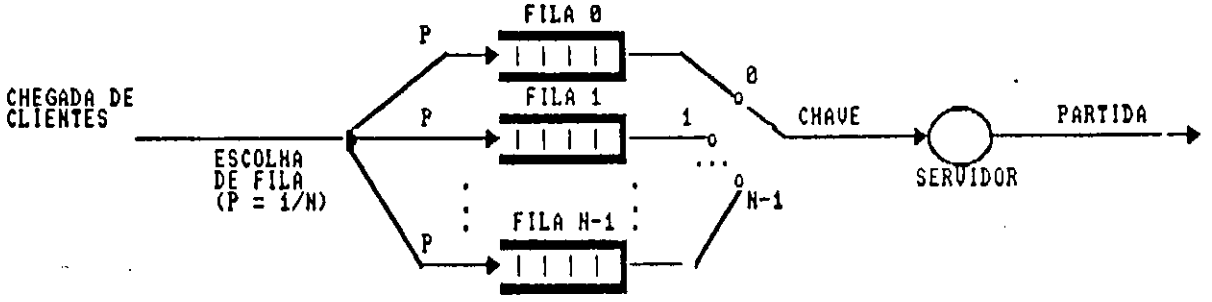


Figura 5.11 - Modelo de Moura para o Protocolo de Passagem de Ficha

Moura considerou que a rede local possui N usuários, cada um conectado a uma interface. "Cada usuário gera pacotes para a sua interface, que os armazena, por ordem de chegada, enquanto aguarda pela chance de transmiti-los. Os usuários são estatisticamente idênticos no que se refere à geração de pacotes, isto é, o tráfego de pacotes para a sub-rede é homogêneo. Os pacotes têm comprimento fixo em P bits. A capacidade do meio de transmissão é C bit/s. Não há geração de pacotes de reconhecimento nas interfaces" [Moura 86:305].

Moura modelou a suposição de um tráfego homogêneo de pacotes para a sub-rede através de uma fonte e de um mecanismo que escolhe, de forma equiprovável, para cada cliente (pacote) gerado, uma das filas (interface) para receber o cliente.

O modelo de Moura apresenta uma única fonte com tempos de interchegada exponencialmente distribuídos com média 1/Ã segundos.

Para modelar o mecanismo de passagem de ficha foi utilizada uma chave giratória: quando ela estiver na posição 0, o servidor atende ao primeiro cliente na fila 0. Quando este cliente parte, a chave passa à posição 1 para que o servidor atenda ao primeiro cliente na fila 1. "Este comportamento é repetido até a fila **N -1** ser atendida, quando então a chave giratória retorna a posição 0 para reiniciar o ciclo. A mudança da chave de uma posição para a próxima é instantânea, refletindo a suposição de que o tempo para passar a ficha entre duas interfaces é desprezível. Os tempos de serviço no servidor são determinísticos, com valor fixo dado por t-t-anel = P/C segundos" [Moura 86] (t-t-anel é o nome da variável que este autor usa para armazenar o tempo de

transmissão de um pacote no anel).

- Modelo no SIM/SAVAD.

A figura 5.12 apresenta o modelo para o Protocolo de Passagem de Ficha usando os elementos do SIM/SAVAD.

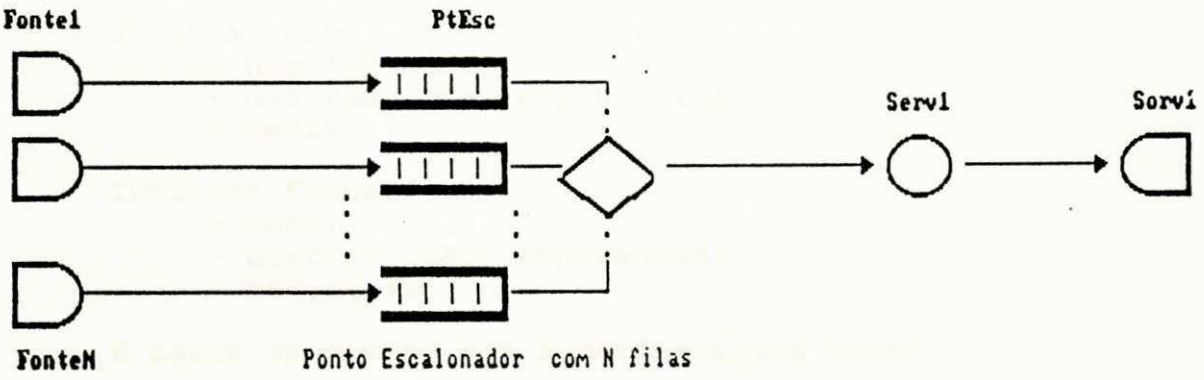


Figura 5.12 - Modelo do SIM/SAVAD para o Protocolo de Passagem de Ficha

Nesse modelo cada usuário da rede é modelado por uma fonte; as interfaces e o protocolo de acesso são modelados por um ponto escalonador. Cada fila modela uma interface e o protocolo de acesso é modelado pela disciplina de escalonamento do ponto escalonador. Para modelar o mecanismo de passagem de ficha escolheu-se um ponto escalonador do tipo cíclico.

Uma estação de serviço, do tipo servidor único com fila de comprimento 1, modela o meio de transmissão. Assim, o ponto escalonador ficará bloqueado quando a estação de serviço estiver atendendo um cliente, isto é, quando o meio de transmissão estiver ocupado.

Para que as N fontes no SIM/SAVAD correspondam a fonte única de Moura, essas fontes foram definidas com tempos de interchegadas exponencialmente distribuídos com média N/Ã segundos (trata-se de uma rede com tráfego homogêneo).

- Definição de Elementos no SIM/SAVAD

O modelo de Moura considera uma rede com 16 interfaces, pacotes de comprimento fixo de 1000 bits, capacidade do meio de transmissão de 1 Mbit/s e tempo médio de interchegada de pacotes de 2,5 milissegundos.

No modelo do SIM/SAVAD são, então, definidas 16 fontes (cada uma com tempo médio de interchegada de 40 milissegundos), um ponto escalonador com 16 filas, uma estação de serviço e 16 rotas. A seguir, apresentam-se os elementos desse modelo considerando uma uts igual a 1 milissegundo.

```
Primeira Fonte:
     - nome: Fonte1
     - distribuição: exponencial
     - média: 40

Segunda Fonte:
     - nome: Fonte2
     - distribuição: exponencial
     - média: 40

Terceira Fonte:
     - nome: Fonte3
     - distribuição: exponencial
     - média: 40

E assim em diante até a décima-sexta fonte.

Estação de Serviço:
     - nome: Serv1
     - comprimento de fila: 1
     - servidor: único
     - distribuição: determinística
     - valor: 1

Ponto Escalonador:
     - nome: PtEsc
     - número de filas: 16
     - comprimento das filas: 100
     - tipo: ciclico
     - ciclo: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16.

Sorvedouro:
     - nome: Sorv1

Classe:
     - nome: c1
     - prioridade: 0

Primeira Rota:
     - nome: rota1
     - classe: c1
     - sequência de nodos:
          Fonte1 >> PtEsc.1 >> Serv1 >> Sorv1

Segunda Rota:
     - nome: rota2
     - classe: c1
     - sequência de nodos:
          Fonte2 >> PtEsc.2 >> Serv1 >> Sorv1

Terceira Rota:
     - nome: rota3
     - classe: c1
```

- sequência de nodos:
Fonte3 >> PtEsc.3 >> Serv1 >> Sorv1

E assim em diante até a décima-sexta rota.

- Resumo dos Resultados

As tabelas 30 e 31 apresentam os resultados obtidos para a utilização do servidor e o tempo médio de espera nas filas do ponto escalonador. Foram consideradas 16000 uts, 7 execuções e nível de confiança igual a 90%. As sementes definidas para os geradores de números aleatórios foram as seguintes: 13, 31051, 2749, 24851, 6967, 31327, 10457.

Esses resultados podem ser comparados com aqueles apresentados em [Moura 86] para as seguintes medidas de desempenho:

- Utilização do servidor: 0.4022 (± 0,98% com nível de confiança de 90%).

- Tempo médio de espera nas interfaces: 0,33691 milissegundos (± 0,99% com nível de confiança de 90%).

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.41015	.....	.....	.....	.....
2	0.41165	0.4108959	0.00106	6.31400	1.15238
3	0.41286	0.4115503	0.00136	2.92000	0.55671
4	0.41267	0.4118305	0.00124	2.35300	0.35516
5	0.41392	0.4122487	0.00143	2.13200	0.32981
6	0.41498	0.4127046	0.00170	2.01500	0.33789
7	0.41730	0.4133605	0.00233	1.94300	0.41310

Tabela 30 - Utilização do Servidor para o Modelo do Protocolo de Passagem de Ficha

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1+	0.34241	.....	.....	.....	.....
2	0.33696	0.3396831	0.00386	6.31400	5.07062
3	0.34044	0.3399363	0.00276	2.92000	1.37025
4	0.33924	0.3397630	0.00228	2.35300	0.79033
5	0.34227	0.3402639	0.00227	2.13200	0.63661
6	0.35932	0.3434403	0.00804	2.01500	1.92615
7	0.35245	0.3447271	0.00809	1.94300	1.72386

Tabela 31 - Tempo Médio de Espera nas Interfaces para o Modelo do Protocolo de Passagem de Ficha

5.2.1.2 - Modelo do Protocolo CSMA/CD

O modelo do Protocolo de Passagem de Ficha, mostrado na Subseção anterior, pode ser facilmente modificado para representar o Protocolo CSMA/CD . Isso é feito alterando-se a disciplina de escalonamento do ponto escalonador para a disciplina do tipo livre e fornecendo-se uma distribuição para o tempo de retransmissão, no caso da ocorrência de colisões de pacotes.

- Modelo de Moura

Em [Moura 86:327] são apresentados os seguintes parâmetros para esse modelo:

- capacidade do barramento : 3 Mbit/s
- comprimento médio das mensagens : 1000 bits
- comprimento do cabeçalho : 100 bits
- comprimento do pacote de dados : 4000 bits
- comprimento do pacote de reconhecimento : 100 bits
- atraso de propagação do meio de transmissão: 0.005 ms

O tempo de serviço definido para o servidor está associado ao tempo de transmissão de um pacote na sub-rede, que é igual a divisão do comprimento do pacote de dados pela capacidade do meio de transmissão.

No modelo de Moura o comprimento de uma mensagem varia de 1 bit a 8000 bits e a função de Retirada Exponencial Binária dobra a média do atraso a cada colisão. Nesse modelo algumas suposições são feitas, destacando-se as seguintes:

a) "As mensagens são segmentadas em pacotes.

b) A cada pacote é adicionado um cabeçalho contendo endereços das interfaces remetentes e destinatária, tipo de pacote, etc.

c) Para cada pacote de dados transmitido, um reconhecimento é preparado e enviado à interface remetente; não há reconhecimento para pacotes de reconhecimento e estes não são incluídos nos de dados."

- Modelo no SIM/SAVAD

A versão atual do SIM/SAVAD não permite uma representação ideal para as suposições destacadas no modelo de Moura, por isto suposições adicionais foram feitas.

Como pode-se observar, em média, uma mensagem forma um pacote de 1100 bits (cabeçalho mais comprimento médio da mensagem).

Observa-se, também, que caso a transmissão de um pacote de reconhecimento seja feita imediatamente a chegada de um pacote de dados, pode-se modelar um pacote como tendo 1200 bits em média.

Essas suposições são aceitáveis numa rede de computadores onde a vazão de mensagens é baixa, o atraso de geração do pacote de reconhecimento seja ínfimo e a janela de transmissão seja de comprimento 1. Nesse caso, pode-se considerar que o intervalo entre a transmissão do pacote de dados e a transmissão do pacote de reconhecimento correspondente é tão pequeno que se pode modelar os dois pacotes como um único pacote de comprimento igual a soma de cada um desses pacotes.

Para um meio que transmite 3 Mbit/s, 1200 bits representam um tempo de transmissão de 0,4 milissegundos.

Quanto ao tempo médio de interchegada, para uma vazão de 300.000 bit/s, a taxa de interchegada de blocos de 1200 bits é de 4 milissegundos.

- Definição de Elementos no SIM/SAVAD

A seguir, apresentam-se os elementos desse modelo, considerando uma uts igual a 1 milissegundo:

Primeira Fonte:
- nome: Fonte1
- distribuição: exponencial
- média: 64

Segunda Fonte:
- nome: Fonte2
- distribuição: exponencial
- média: 64

Terceira Fonte:
- nome: Fonte3
- distribuição: exponencial
- média: 64

E assim em diante até a décima-sexta fonte, lembrando que o tempo médio de interchegada foi multiplicado pelo número de fontes.

Estação de Serviço:
- nome: Serv1
- comprimento de fila: 1
- servidor: único
- distribuição: exponencial
- média: 0.4

Ponto Escalonador:
- nome: PtEsc
- número de filas: 16
- comprimento das filas: 100
- tipo: livre
- distribuição: exponencial
- média: 0.01

Sorvedouro:
- nome: Sorv1

Classe:
- nome: c1
- prioridade: 0

Primeira Rota:
- nome: rota1
- classe: c1
- sequência de nodos:
  Fonte1 >> PtEsc.1 >> Serv1 >> Sorv1

Segunda Rota:
- nome: rota2
- classe: c1
- sequência de nodos:
  Fonte2 >> PtEsc.2 >> Serv1 >> Sorv1

Terceira Rota:
- nome: rota3
- classe: c1
- sequência de nodos:
  Fonte3 >> PtEsc.3 >> Serv1 >> Sorv1

E assim em diante até a décima-sexta rota.

- Resumo dos Resultados

A tabela 32 apresenta os resultados obtidos para o atraso médio de transmissão de pacotes. Foram consideradas 25000 uts, 7 execuções e nível de confiança igual a 90%. As sementes definidas para os geradores de números aleatórios foram as seguintes: 13, 31051, 2749, 24851, 6967, 31327, 10457, 14479, 29983, 7757.

Cada um destes resultados pode ser comparado com a média obtida no modelo de Moura para o protocolo CSMA/CD, que é o seguinte:

- Atraso Médio para Transmitir um Pacote: 0.44 ms

Neste caso o resultado, devido a baixa vazão da rede, pode também ser comparado com o resultado obtido num sistema M/M/1.

Em [Moura 86:328] é analisado o efeito de outras vazões numa rede com CSMA/CD, sendo demonstrado que a partir de uma vazão normalizada de 0.4, a rede entra em colapso. No caso do SIM/SAVAD os resultados foram satisfatórios até uma vazão normalizada de 0.3, devido às simplificações que foram utilizadas.

Rodada (n)	Resultado	Media Amostral	Desvio Padrao	Tabela t-student	Interv. de Confianca (percent.)
1	0.44135	.....	.....	.....	.....
2	0.43956	0.4404512	0.00127	6.31400	1.28240
3	0.43599	0.4389640	0.00273	2.92000	1.04724
4	0.44685	0.4409356	0.00453	2.35300	1.20825
5	0.44595	0.4419389	0.00452	2.13200	0.97474
6	0.44308	0.4421296	0.00407	2.01500	0.75687
7	0.44957	0.4431930	0.00466	1.94300	0.77200

Tabela 32 - Tempo Médio de Vida dos Clientes no Modelo do Protocolo CSMA/CD

CAPITULO 6

CONCLUSAO

C A P I T U L O 6

CONCLUSÕES E SUGESTÕES

O SIM/SAVAD é um simulador de modelos de redes de filas que pode representar sistemas que exibem contenção de recursos. Ele faz parte integrante do módulo solução do Sistema de Avaliação de Desempenho (SAVAD).

Os elementos de modelagem do SIM/SAVAD permitem modelar com facilidade uma grande variedade de sistemas como, por exemplo, sistemas de computadores, de redes de computadores e de tráfego.

O SIM/SAVAD é particularmente útil na modelagem de redes de computadores. Dois dos seus elementos de modelagem ponto de sincronização e ponto escalonador são, respectivamente, adequados à modelagem do mecanismo de controle de fluxo de redes de computadores e à modelagem de protocolos de acesso ao meio em redes locais de computadores.

A utilização do paradigma de objetos no desenvolvimento do SIM/SAVAD possibilitou um projeto modular, extensível e reutilizável. Os mecanismos de herança e amarração dinâmica suportados pela linguagem de programação C++, usada na implementação do SIM/SAVAD, foram facilidades desse paradigma também exploradas.

A experiência adquirida com o uso do paradigma de objetos na construção do SIM/SAVAD vem contribuir no desenvolvimento científico do projeto orientado a objetos e da programação orientada a objetos, especialmente aplicados ao projeto e à implementação de programas simuladores.

A defesa dessa dissertação encerra uma etapa do desenvolvimento do SIM/SAVAD. Para dar continuidade a essa etapa seguem algumas sugestões de pesquisas:

1) Aprimorar a apresentação das medidas de desempenho obtidas pelo SIM/SAVAD.

2) Aprimorar integração do SIM/SAVAD com os outros módulos do SAVAD.

3) Permitir ao usuário uma forma mais simples para especificação de modelos considerados básicos, exemplo, sistemas A/B/m/K/M.

4) Permitir ao usuário a especificações de modelos através de uma interface gráfica.

5) Portar o SIM/SAVAD para que possa ser utilizado sob outros sistemas operacionais, como por exemplo o UNIX e para que possa ser utilizado a partir de ambientes como o Windows da Microsoft.

6) Ampliar os parâmetros que definem a simulação de um modelo. Foram relacionadas algumas sugestões:

a) permitir o limite de simulação por número de eventos.

b) permitir a definição precisa da semente que será associada a cada gerador de números aleatórios, em cada uma das simulações que forem feitas.

7) Estender os elementos do SIM/SAVAD para permitir a modelagem de uma variedade maior de sistemas de redes de filas.

8) Estender a funcionalidade dos elementos de modelagem do SIM/SAVAD. Foram relacionadas algumas sugestões:

a) tornar o ponto de duplicação um ponto de multiplicação, isto é, permitir que um cliente que chegue neste nodo gere múltiplos clientes.

b) permitir outras disciplinas de escalonamento de clientes em fila. Como exemplos: LCFS ("Last Come First Served") e HOL ("Header of Line").

c) permitir outras opções de Funções de Distribuição de Probabilidade, como por exemplo a Distribuição Normal.

d) permitir roteamento probabilístico conforme especificado na interface do SIM/SAVAD.

e) atendimento das estações de serviço conforme classes diferentes de clientes.

APENDICE

PROTOCOLO DE CLASSES

Vlimites: armazena o limite, em tempo simulado, para o término da simulação.

VlistaDeNPAs: apontador para um objeto da classe ListaDeNPAs.

6. MÉTODOS

6.1 - PÚBLICOS:

Inicializa: gerencia a fase de inicialização do simulador.

InicializaAmbiente: inicializa as listas.

InicializaModelo: recebe a especificação do modelo e faz as inicializações necessárias.

CriaElementos: lê a definição recebida pela interface e armazena-a na forma conveniente para simulação.

NodosDaRota: para cada rota definida na interface faz sua conversão num objeto da classe ListaDeNodos.

PercorreLista: verifica nos objetos da classe ListaDeNodo se há um nodo que tenha um determinado nome.

TestaDesbloqueioNoModelo: verifica, nos casos onde VclientesDesbloqueados e VeventosFuturo são listas vazias, a possibilidade de desbloquear algum cliente para resolver impasses.

InicializaFinalizacao: gerencia métodos EspecificaEstatisticas e EspecificaLimites.

EspecificaLimites: recebe e armazena valor de Vlimites.

IniRotas_Nos: percorre rotas preparando seus nodos para a simulação.

Executa: gerencia a simulação.

TrataDesbloqueio: envia mensagem para o nodo bloqueado onde ocorreu um desbloqueio, e envia mensagem para o nodo bloqueador receber o(s) cliente(s) que estiver(em) sendo desbloqueado(s).

TrataMovimentacaoDeCliente: prepara o avanço de um cliente.

TrataClienteBloqueado: envia mensagem para o nodo bloqueador que causou um bloqueio.

RelataEstatistica: gerencia a fase de apresentação de resultados.

GeradorVirtual: cria um objeto de uma das seguintes classe: GeradorExponencial, GeradorUniforme, GeradorDeterministico.

EscalonadorVirtual: cria um objeto de uma das seguintes classes: EscalonadorCiclico, EscalonadorLivre e EscalonadorRandomico.

CriaArquivos: abre arquivo para armazenar os resultados da simulação.

IniPtControle: aciona métodos que enviarão mensagens para objetos das classes que representam nodos que possuem mais de uma entrada e/ou mais de uma saida.

IndicaSemente: fornece as sementes utilizadas nos objetos da classe GeradorDeNPA armazenados em VlistaDeNPAs.

1. **CLASSE:** Bloqueado

2. **DESCRIÇÃO:** elemento da lista definida na classe ListaDesBloqueioAnterior

3. **SUPERCLASSES:** Não há

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:**

(pública):

Vbloqueado: apontador para um cliente ou um nodo (ponto de controle)

Vtipo: indicador do tipo do elemento apontado por Vapontador.

VnumDeClientesSaindo: armazena número de clientes bloqueados em um nodo.

6. **MÉTODOS**

6.1 - PÚBLICOS:

Bloqueado: construtor.

InformaVtipo: retorna Vtipo.

InformaVnumDeClientesSaindo: retorna VnumDeClientesSaindo.

InformaBloqueado: retorna Vbloqueado.

1. **CLASSE:** Classe

2. **DESCRIÇÃO:** encapsula as caracteristicas de uma classe de clientes

3. **SUPERCLASSES:** Não há

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:**

(pública):

Vprioridade: armazena a prioridade da classe de clientes.

Vrotas: apontador para um objeto da classe ListaDeRotas.

Vnome: armazena nome da classe de clientes.

6. **MÉTODOS**

6.1 - PÚBLICOS:

Classe: construtor.

DaPrioridade: informa prioridade da rota.

AcrescentaRota: recebe uma rota para armazenar em Vrotas.

AtualizaEstatistica: coleta dados para as medidas de desempenho.

RelataEstatistica: apresenta medidas de desempenho coletadas.

1. **CLASSE:** Cliente

2. **DESCRIÇÃO:** encapsula as caracteristicas de um cliente.

3. **SUPERCLASSES:** Não há

4. **VARIÁVEIS DE CLASSE:**

VtotalGerado: contador do numero de clientes gerados.

VtotalSaida: contador de clientes que sairam do modelo.

VatualMax: registra o numero maximo de clientes que o modelo chegou a possuir.

5. **VARIÁVEIS DE INSTÂNCIA:**

(pública):

Vnascimento: registra o tempo em que este cliente entrou no modelo.

VmomentoDeInicio: registra o tempo de ocorrência do último evento de um cliente.

VnoAtual: registra o nodo atual aonde esta´o cliente.

Vrota: registra a rota que o cliente segue.

6. **MÉTODOS**

6.1 - PÚBLICOS:

cria : gera instância dessa classe.

EliminaCliente: coleta estatísticas e destrói o cliente.

InformaTempo: informa o tempo armazenado em VmomentoDeInicio.

RegistraTempo: armazena tempo de inicio da última atividade do cliente na variável VmomentoDeInicio.

RegistraNo: registra novo valor em VnoAtual.

InformaNodoAtual: informa o nodo armazenado em VnoAtual.

InformaRota: informa a rota armazenada em Vrota.

Avanco: gerencia o avanço de um cliente na sua rota.

Inicializa: inicializa variáveis de instância.

InicializaClass: inicializa variáveis de classe.

RegistraNascimento: armazena valor em Vnascimento.

RelataEstatistica: informa estatísticas que podem ser obtidas a partir das variáveis de classe.

1. **CLASSE:** Desbloqueado

2. **DESCRIÇÃO:** elemento da lista definida na classe ListaDeDesbloqueados

3. **SUPERCLASSES:** Não há

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:**

(pública):

Vcliente: aponta para o cliente que foi desbloqueado e precisa ser processado.

VproxNo: aponta para o nodo que bloqueou o cliente e agora pode desbloqueá-lo.

6. **MÉTODOS**

6.1 - PÚBLICOS:

cria: gera instância dessa classe.

InformaCliente: informa o cliente que está armazenado em Vcliente.

InformaNo: informa o nodo que está armazenado em VnoAtual.

destroi: elimina uma instância.

1. **CLASSE:** EntradaPtFusao.

2. **DESCRIÇÃO:** elemento da lista definida na classe ListaDeEntradaPtFusao. Permite que os pontos de fusão controlem os clientes que ficam bloqueados nos nodos anteriores. Para cada entrada de ponto de fusão, um objeto desta classe é gerado.

3. **SUPERCLASSES:** Não há

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:**

(pública):

Vrota: apontador para uma das rotas que chega ao ponto de fusao.

Vlista: apontador para um objeto da classe ListaDeBloqueio-Anterior.

6. **MÉTODOS**

6.1 - PÚBLICOS:

InformaRota: retorna o valor de Vrota.

InformaListaDeBloqueioAnterior: retorna o valor de Vlista.

1. **CLASSE:** EscalonadorCiclico.

2. **DESCRIÇÃO:** faz o escalonamento de filas segundo a variável Vciclo

3. **SUPERCLASSES:** EscalonadorDeFilaVirtual

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:**

Vciclo: apontador para um objeto da classe ListaCicloDoPtEscalonador.

VfilaRotaEscalonada: apontador para um objeto da classe Fila_Rota.

VliberacoesFeitas: conta o número de clientes que já foram escalonados da fila da vez.

6. **MÉTODOS**

6.1 - PÚBLICOS:

EscalonaFila: a partir de um objeto da classe VezNoCiclo, que é obtido a partir da variável Vciclo, localiza uma fila para ter o seu primeiro cliente escalonado. A fila é obtida a partir de objeto da classe ListaDeFilas_Rotas fornecido como parâmetro deste método .

Inicializa: inicializa as variáveis de instância.

1. **CLASSE:** EscalonadorDeFilaVirtual.

2. **DESCRIÇÃO:** classe abstrata definida para encapsular funções virtuais (Zortech, 1990) que serão redefinidos nas suas sub-classes.

3. **SUPERCLASSES:** Não há

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTANCIA:** Não há

6. **MÉTODOS**

6.1 - PÚBLICOS:

(virtual) EscalonaFila: só tem significado na classe derivada que o redefine.

1. **CLASSE:** EscalonadorLivre.

2. **DESCRIÇÃO:** faz escalonamento de filas detectando clisões.

3. **SUPERCLASSES:** EscalonadorDeFilaVirtual

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:**

VfilaRotaEscalonada: apontador para um objeto da classe Fila_Rota.

Vgerador: apontador para um objeto de uma das subclasses da classe GeradorAmostraVirtual. Utilizado nos casos em que ocorrem colisão.

6. **MÉTODOS**

6.1 - PÚBLICOS:

EscalonaFila: no caso de não detectar uma colisão, este método indica a fila que pode liberar um cliente.

1. **CLASSE:** EscalonadorRandomico.

2. **DESCRIÇÃO:** seleciona uma fila para escalonamento de forma randômica.

3. **SUPERCLASSES:** EscalonadorDeFilaVirtual

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:**

VfilaRotaEscalonada: apontador para um objeto da classe Fila_Rota.

Vgerador: apontador para um objeto da classe GeradorNPA. Utilizado para indicar a fila que deve ser selecionada.

6. **MÉTODOS**

6.1 - PÚBLICOS:

EscalonaFila: usando Vgerador, indica uma fila para ter seu primeiro cliente escalonado.

1. **CLASSE:** EstacaoDeServico

2. **DESCRIÇÃO:** representa um nodo do tipo estação de serviço com número limitado de servidores.

3. **SUPERCLASSES:** NodoVirtual

4. **VARIÁVEIS DE CLASSE:**

(pública):

Vtipo: indica que o objeto é do tipo EstacaoDeServico.

5. **VARIÁVEIS DE INSTÂNCIA:**

(pública):

Vgerador: apontador para um objeto de classe derivada da classe GeradorVirtual.

VflagDeDisponibilidade: indica se este nodo está no estado disponível.

VnumServ: informa o número de servidores que a instância possui.

VnumServLivres: informa o número de servidores que estão livres.

Vfila: apontador para uma objeto da classe Fila ou FilaInfinita.

VflagFila: indica qual tipo de fila que está sendo utilizado a partir de Vfila.

VnumDeClientesQueVisitaram: contador do número de clientes que entraram na estação de serviço.

VtempoDasVisitas: acumula o tempo que os clientes ficam na instância.

VtempoDeBloq: acumula o tempo que os clientes ficam bloqueados na instância.

VnumClientesBloqueados: acumula o número de clientes que ficaram bloqueados na instância.

VnumServBloqueados: indica o número de servidores bloqueados num determinado tempo.

6. MÉTODOS

6.1 - PÚBLICOS:

cria: gera instância dessa classe.

EscalonaCliente: escalona serviço.

VisitaNo: processa a visita de um cliente a estação de serviço. Ativa o método EscalonaCliente ou o método VisitaNo na fila apontada por Vfila.

Disponibilidade: informa a disponibilidade da fila mais o número de servidores livres.

EstadoDisponivel: informa se este nodo está no estado disponível.

Saida: processa a saída de um cliente da estação de serviço.

InsereClienteBloqueado: armazena informação a respeito de bloqueio em nodos anteriores.

TestaDesbloqueioEmNoAnterior: verifica se há nodo anterior que pode ter cliente(s) desbloqueado(s). Este método deve ser ativado nos momentos que o método Saida for ativado.

TrataClienteBloqueado: faz o bloqueio de um cliente na estação de serviço.

TrataDesbloqueio: processa o avanço de um cliente que estava bloqueado na estação de serviço.

RelataEstatistica: informa medidas de desempenho coletadas.

1. **CLASSE:** EstacaoDeServicoInfinito

2. **DESCRIÇÃO:** representa um nodo do tipo estação de serviço com infinito servidores.

3. **SUPERCLASSES:** EstacaoDeServico

4. **VARIÁVEIS DE CLASSE:** não há

5. **VARIÁVEIS DE INSTÂNCIA:**

(pública):

VnumServOcupados: indica o número de servidores ocupados num determinado momento, portanto serve para indicar o número de clientes sendo atendidos neste momento.

6. **MÉTODOS**

6.1 - PÚBLICOS:

EstacaoDeServicoInfinito: construtor.

EscalonaCliente: escalona serviço.

Disponibilidade: informa que sempre há servidores disponíveis nesta instância.

EstadoDisponivel: informa que esta instância sempre está no estado disponível.

Saida: processa a saída de um cliente da estação de serviço.

TrataClienteBloqueado: faz o bloqueio de um cliente na estação de serviço.

TrataDesbloqueio: processa o avanço de um cliente que estava bloqueado na estação de serviço.

1. **CLASSE:** Evento

2. **DESCRIÇÃO:** elemento da lista definida na classe ListaDeEventos. Representa um evento que foi escalonado para ocorrer num tempo futuro

3. **SUPERCLASSES:** Não há

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTANCIA:**

(pública):

Vtempo: armazena o tempo de ocorrência do objeto EventoTipo.

Vcliente: apontador para um objeto da classe Cliente. Aponta para o cliente relacionado com o evento futuro.

6. **MÉTODOS**

6.1 - PÚBLICOS:

cria: gera uma instância dessa classe.

destroi: elimina uma instância

EventoTipo: construtor.

InformaTempo: informa tempo aramazenado em Vtempo.

InformaCliente: retorna apontador para cliente armazenado em Vcliente.

1. **CLASSE:** Fila

2. **DESCRIÇÃO:** armazena instâncias da classe Cliente que necessitam esperar algum evento.

3. **SUPERCLASSES:** Não há

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:**

(pública):

VlistaDeClientes: armazena um apontador para um objeto da classe ListaDeClientes. Este objeto armazenará os clientes que estiverem na fila.

VListaDeBloqueioAnterior: apontador para um objeto da classe ListaDeBloqueioAnterior.

Vlimite: armazena o número máximo de clientes que podem ser armazenados.

6. **MÉTODOS**

6.1 - PÚBLICOS:

cria: gera uma instância dessa classe.

Inicializa: inicializa variáveis de instância.

RelataEstatistica: informa medidas de desempenho coletadas.

TamanhoAtual: informa quantos clientes há em VlistaDeClientes.

Disponibilidade: informa a diferença entre Vlimite e o valor informado pelo método TamanhoAtual, ou seja, a quantidade de clientes que a fila pode receber num determinado momento.

TestaDesbloqueioEmNodoAnterior: verifica se há nodo anterior que foi desbloqueado. Este método deve ser ativado nos momentos que VcompAtual é decrementado.

VisitaNo: processa a chegada de um cliente na fila.

InsereClienteBloqueado: armazena informação sobre cliente que ficou bloqueado no nodo anterior.

Saida: processa a saída de um cliente da fila.

PegaDaFila: retira o primeiro cliente da fila segundo uma disciplina de escalonamento de clientes. Usa métodos SelecionaCliente e Saida.

SelecionaCliente: indica o primeiro cliente da fila segundo uma disciplina de escalonamento de clientes.

RetiraAteNClientes: retira até n clientes da fila.

InformaSeVazia: verifica se VlistaDeClientes está vazia.

IndicaPrimeiroCliente: retorna o primeiro cliente da fila segundo sua disciplina de atendimento, por exemplo FCFS.

1. **CLASSE:** FilaInfinita

2. **DESCRIÇÃO:** armazena instâncias da classe Cliente que necessitam esperar algum evento.

3. **SUPERCLASSES:** Fila

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:** Não há

6. **MÉTODOS**

6.1 - PÚBLICOS:

FilaInfinita: construtor.

Disponibilidade: informa que esta fila está sempre disponível, pois não tem limite de comprimento.

VisitaNo: processa a chegada de um cliente na fila.

1. **CLASSE:** Fila_Rota

2. **DESCRIÇÃO:** armazena um apontador para uma das filas de um ponto escalonador, ou ponto de sincronização e outro apontador para a rota associada.

3. **SUPERCLASSES:** Não há

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTANCIA:**

(pública):

Vfila: armazena um apontador para um objeto da classe Fila.

Vrota: armazena um apontador para um objeto da classe Rota.

VflagDeDisponibilidade: indica se a fila pode receber clientes.

6. **MÉTODOS**

6.1 - PÚBLICOS:

InformaFila: retorna o valor de Vfila.

InformaFlagDeDisponibilidade: retorna o valor de VfladDeDisponibilidade.

AtualizaFlagDeDisponibilidade: atualiza o valor de VfladDeDisponibilidade.

InformaRota: retorna o valor de Vrota.

RegistraRota: atribui valor a Vrota.

1. **CLASSE:** Fonte

2. **DESCRIÇÃO:** representa nodos do tipo fonte, isto é, geradores de clientes baseados em tempos de interchegadas definidos por uma distribuição de probabilidade.

3. **SUPERCLASSES:** NodoVirtual

4. **VARIÁVEIS DE CLASSE:**

(pública):

Vtipo: indica que o objeto é do tipo fonte.

5. **VARIÁVEIS DE INSTÂNCIA:**

(pública):

Vgerador: apontador para um objeto de classe derivada da classe GeradorVirtual.

Vrota: apontador para um objeto da classe Rota. A fonte deve estar associada a uma rota.

VtempoUltimaSaida: registra o tempo de saída do cliente que foi gerado por último.

VclientesGerados: registra o número de clientes gerados.

VclientesBloqueados: registra o número de clientes desta fonte que tiveram sua saída bloqueada.

6. **MÉTODOS**

6.1 - PÚBLICOS:

cria: gera uma instância dessa classe.

InicializaFonte: inicializa variável Vrota e ativa método AtivaFonte.

EscalonaCliente: escalona geração de cliente.

AtivaFonte: gera um cliente e ativa o método EscalonaCliente.

DizTipo: retorna o valor de Vtipo.

TrataClienteBloqueado: ativa método EscalonaCliente para reescalonar um cliente que foi bloqueado.

Saida: processa o avaço de um cliente gerado.

1. **CLASSE:** GeradorDeterministico

2. **DESCRIÇÃO:** retorna um valor constante para um objeto que precise de amostras.

3. **SUPERCLASSES:** GeradorVirtual

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:** não há

6. **MÉTODOS**

   6.1 - PÚBLICOS:

   Amostra: retorna um valor constante

1. **CLASSE:** GeradorExponencial

2. **DESCRIÇÃO:** retorna amostras de uma Função Distribuição Exponencial

3. **SUPERCLASSES:** GeradorVirtual

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:** não há

6. **MÉTODOS**

   6.1 - PÚBLICOS:

   Amostra: gera e retorna uma amostra aleatória.

1. **CLASSE:** GeradorUniforme

2. **DESCRIÇÃO:** retorna amostras de uma Função Distribuição Uniforme.

3. **SUPERCLASSES:** GeradorVirtual

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:** não há

6. **MÉTODOS**

   6.1 - PÚBLICOS:

   Amostra: gera e retorna uma amostra aleatória.

1. **CLASSE:** GeradorVirtual

2. **DESCRIÇÃO:** classe abstrata definida para encapsular funções virtuais (Zortech, 1990) que serão redefinidos nas suas sub-classes.

3. **SUPERCLASSES:** Não há

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTANCIA:** não há

6. **MÉTODOS**

6.1 - PÚBLICOS:

(virtual) Amostra: definido para tornar possivel a sua redefinição nas classes derivadas.

1. **CLASSE:** ListaCicloPtEscalonador.

2. **DESCRIÇÃO:** armazena elementos da classe VezNoCiclo.

3. **SUPERCLASSES:** zDList

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:** Não ha´

6. **MÉTODOS**

   6.1 - PÚBLICOS:

   InformaVezNoCiclo: retorna elemento da classe VezNoCiclo.

1. **CLASSE:** ListaDeBloqueioAnterior

2. **DESCRIÇÃO:** armazena elementos da classe Bloqueado que vão sendo gerados a medida que os clientes tentam avançar pelo modelo.

3. **SUPERCLASSES:** zDList

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:** Não ha´

6. **MÉTODOS**

6.1 - PÚBLICOS:

cria: gera uma instância dessa classe.

InsereBloqueado: armazena objeto da classe Bloqueado na lista

InformaBloqueado: retorna objeto da classe Bloqueado

RetiraBloqueado: retira objeto da classe Bloqueado que estava na lista.

InformaSeVazia: verifica se a lista está vazia.

1. **CLASSE:** ListaDeClasses

2. **DESCRIÇÃO:** armazena objetos da classe Classe

3. **SUPERCLASSES:** zDList

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:** Não há

6. **MÉTODOS**

   6.1 - PÚBLICOS:

   InsereClasse: armazena um objeto da classe Classe.

   RelataEstatistica: envia a mensagem RelataEstatistica para os objetos da classe Classe que estiverem armazenados.

1. **CLASSE:** ListaDeClientes

2. **DESCRIÇÃO:** armazena clientes

3. **SUPERCLASSES:** zDList

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:** Não há

6. **MÉTODOS**

   6.1 - PÚBLICOS:

   cria: gera uma instância dessa classe.

   destroi: elimina uma instância

   InsereCliente: armazena um cliente na lista.

   GeraNClientes: retorna lista com n clientes.

   InformaSeVazia: informa se há algum cliente na lista.

   RetiraCliente: retira um cliente da lista.

1. **CLASSE:** ListaDeDesbloqueados

2. **DESCRIÇÃO:** armazena elementos da classe Desbloqueio que foram gerados com o teŕmino de algum bloqueio.

3. **SUPERCLASSES:** zDList

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:** Não há

6. **MÉTODOS**

6.1 - PÚBLICOS:

InsereDesbloqueado: armazena um cliente que estava bloqueado e o nodo para onde ele se dirige.

PegaDesbloqueio: retorna nodo se há um objeto da classe Desbloqueio na lista.

PegaCliente: retorna cliente se há um objeto da classe Desbloqueio na lista.

destroi: elimina uma instância.

1. **CLASSE:** ListaDeEntradaPtFusao

2. **DESCRIÇÃO:** armazena objeto da classe EntradaPtFusao conforme o número de entradas do ponto de fusão que estiver instanciando esta classe

3. **SUPERCLASSES:** zDList

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:** Não há

6. **MÉTODOS**

6.1 - PÚBLICOS:

InsereEntradaParaFusao: armazena objeto da classe EntradaPt-Fusao.

SatisfazCondicaoComRota: verifica se com a chegada de um cliente de determinada rota, há condições para realizar uma fusão. Este método é ativado pelo método EstadoDisponivel da classe PontoDeFusão.

CondicaoSatisfeita: verifica se há clientes para fazer uma fusão.

SatisfazNovaFusao: verifica se com a chegada de um cliente de determinada rota, há condições para realizar mais uma fusão.

InformaEntrada: retorna um dos objetos da classe Entrada-PtFusao estão armazenados.

AchaEntradaPelaRota: encontra objeto da classe EntradaPtFusao associada com uma determinada rota.

RegistraRota: faz que um dos objetos da classe EntradaPtFusao armazene uma das rotas que chegam ao ponto de fusão.

1. **CLASSE:** ListaDeEventos

2. **DESCRIÇÃO:** armazena objetos da classe EventoTipo

3. **SUPERCLASSES:** zDList

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTANCIA:** Não há

6. **MÉTODOS**

   6.1 - PÚBLICOS:

   cria: gera uma instância dessa classe.

   destroi: elimina uma instância

   InsereEvento: insere um objeto EventoTipo na lista segundo seu tempo de ocorrência.

   RemoveEvento: retira o primeiro objeto EventoTipo da lista.

   InsereAntes: insere um objeto EventoTipo na lista após o elemento corrente.

   RelataEstatistica: apresenta o número de eventos que estão na lista.

1. **CLASSE:** ListaDeFilas_rotas

2. **DESCRIÇÃO:** armazena objetos da classe Fila_Rota conforme o número de filas e rotas que chegam a nodos do tipo ponto escalonador ou ponto de sincronização.

3. **SUPERCLASSES:** zDList

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:** Não há

6. **MÉTODOS**

6.1 - PÚBLICOS:

AchaFilaPeloNumero: retorna um objeto da classe Fila_Rota onde a fila estiver associada a um determinado número.

AchaFilaPeloRota: retorna um objeto da classe Fila_Rota onde a fila estiver associada a uma determinada rota.

AchaNumDaFilaPelaFila: dada uma fila retorna o número dela.

VisitaNo: coloca um cliente que chega em um nodo em uma de suas filas.

RegistraRota: associa uma rota com uma fila.

RelataEstatistica: envia a mensagem RelataEstatistica para as filas armazenadas nos objetos da classe Fila_Rota.

1. **CLASSE:**ListaDeNodos

2. **DESCRIÇÃO:** armazena os nodos do modelo simulado.

3. **SUPERCLASSES:** zDList

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTANCIA:** Não há

6. **MÉTODOS**

   6.1 - PÚBLICOS:

   cria: gera uma instância dessa classe.

   destroi: elimina uma instância

   RelataEstatistica: envia a mensagem RelataEstatistica para os nodos.

   InsereNodo: armazena um nodo.

1. **CLASSE:** ListaDeNPA.

2. **DESCRIÇÃO:** armazena objeto da classe GeradoNPA.

3. **SUPERCLASSES:** zDList

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:** Não ha´

6. **MÉTODOS**

   6.1 - PÚBLICOS:

   InsereGeradoNPA: armazena um objeto da classe GeradorNPA.

   InformaGeradorNPA: retorna um objeto da classe GeradorNPA.

1. **CLASSE:** ListaDeRotas

2. **DESCRIÇÃO:** armazena objetos da classe Rota

3. **SUPERCLASSES:** zDList

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:** Não há

6. **MÉTODOS**

   6.1 - PÚBLICOS:

   cria: gera uma instância dessa classe.

   destroi: elimina uma instância

   InsereRota: armazena uma rota.

   RelataEstatistica: envia a mensagem RelataEstatistica para as rotas armazenadas.

1. **CLASSE:** NodoVirtual

2. **DESCRIÇÃO:** classe abstrata definida para encapsular funções virtuais [Zortech 90] que serão redefinidas nas suas subclasses.

3. **SUPERCLASSES:** Não há

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:**

(pública):

Vnome: armazena o nome recebido pela instância que foi criada.

6. **MÉTODOS**

6.1 - PÚBLICOS:

(virtual) Saida: só tem significado na classe derivada que o redefine.

(virtual) TestaDesbloqueioEmNoAnterior: só tem significado na classe derivada que o redefine.

(virtual) VisitaNo: só tem significado na classe derivada que o redefine.

(virtual) Disponibilidade: só tem significado na classe derivada que o redefine.

(virtual) EstadoDisponivel: só tem significado na classe derivada que o redefine.

(virtual) TrataClienteBloqueado: só tem significado na classe derivada que o redefine.

(virtual) TrataDesbloqueio: só tem significado na classe derivada que o redefine.

(virtual) Inicializa: só tem significado na classe derivada que o redefine.

(virtual) TestaIndisponibilidadeMultipla: só tem significado na classe derivada que o redefine.

(virtual) TestaDesbloqueioDeEmergencia: só tem significado na classe derivada que o redefine.

(virtual) RelataEstatistica: só tem significado na classe derivada que o redefine.

(virtual) InsereClienteBloqueado: só tem significado na classe derivada que o redefine.

1. **CLASSE:** PontoDeDuplicacao

2. **DESCRIÇÃO:** representa nodos do tipo ponto de duplicacao, isto e´nodos que ao receberem um cliente, geram um novo cliente.

3. **SUPERCLASSES:** NodoVirtual

4. **VARIÁVEIS DE CLASSE:**

(pública):

Vtipo: indica que o objeto é do tipo ponto de duplicação.

5. **VARIÁVEIS DE INSTÂNCIA:**

(pública):

VclientesGerados: contabiliza a quantidade de clientes que foram gerados.

VnumClientesBloqueados: contabiliza as duplicações bloqueadas.

VtempoDeBloq: contabiliza o tempo que a instância ficou no estado bloqueado.

VmomentoDeBloqueio: indica o instante em que ocorreu o último bloqueio da instância.

VListaDeBloqueioAnterior: apontador para um objeto da classe ListaDeBloqueioAnterior

VproxNodo1 e VproxNodo2: indicam os próximos nodos de um ponto de duplicação.

Vrota1 e Vrota2: apontadores para objetos da classe Rota. Serve para indicar a rota de saída do cliente que chega para uma duplicação e a rota do cliente resultante da duplicação.

6. **MÉTODOS**

6.1 - PÚBLICOS:

VisitaNo: processa a visita de um cliente a um ponto de duplicação. Uma visita só se justifica se estiver associada a uma duplicação, necessitando que se processe a saída do cliente recebido e a do cliente que foi gerado.

EstadoDisponivel: indica se a instância está no estado disponível.

Disponibilidade: retorna o mesmo valor que o método EstadoDisponivel.

Saida: processa a saída do cliente resultante de uma duplicação e do cliente que provocou a duplicação.

InsereClienteBloqueado: armazena informação a respeito de bloqueios em nodos anteriores.

TestaDesbloqueioEmNoAnterior: caso haja uma duplicação bloqueada, verifica se pode efetuá-la.

TrataDesbloqueioDeEmergencia: processa o desbloqueio da instância. Este método possibilita a realização da(s) duplicação(ões) que estava(m) bloqueada(s).

RelataEstatistica: apresenta informações coletadas a partir das variáveis de instância.

1. **CLASSE:** PontoDeFusao

2. **DESCRIÇÃO:** representa nodos do tipo ponto de fusão, isto e´ nodos que recebem clientes e os funde em um único cliente.

3. **SUPERCLASSES:** NodoVirtual

4. **VARIÁVEIS DE CLASSE:**

(pública):

Vtipo: indica que o objeto é do tipo ponto de fusão.

5. **VARIÁVEIS DE INSTÂNCIA:**

(pública):

VcondicaoDeFusao: indica o número de clientes que são necessários para efetuar uma fusão.

VemFusao: indica se está fazendo uma fusão, isto é recebendo clientes através do método VisitaNo.

VclientesRecebidosNoMomento: indica a quantidade de clientes que já foram recebidos para uma fusão. Essa quantidade deve ser comparada com a variável VcondicaoDeFusao.

VnumDeFusoes: contabiliza as fusões que já foram realizadas.

VnumFusoesBloqueadas: contabiliza as fusões bloqueadas.

VtempoBloqueado: contabiliza o tempo que o objeto ficou no estado bloqueado.

VmomentoDeBloqueio: indica o tempo em que ocorreu o último bloqueio do objeto.

VflagDeBloq: indica se o objeto está no estado bloqueado.

Ventrada: apontador para um objeto da classe ListaDeEntrada-PtFusao

VproxNodo: indica o proximo nodo de um ponto de fusão.

VrotaDeSaida: apontador para um objeto da classe Rota. Serve para indicar qual a rota do cliente resultante da fusão.

VlistaDeFusao: armazena apontadores para os clientes que são recebidos em uma fusão. Utilizada pelo método VisitaNo.

6. MÉTODOS

6.1 - PÚBLICOS:

VisitaNo: processa a chegada de um cliente a um ponto de fusão. Uma chegada só ocorre se estiver associada a uma fusão, necessitando que ocorram as chegadas dos demais clientes necessários a essa fusão, em um mesmo tempo registrado pelo relógio simulado.

EstadoDisponivel: dada uma rota, indica se o objeto está no estado disponível.

Disponibilidade: retorna o mesmo valor que o método EstadoDisponivel.

Saida: processa a saída do cliente resultante de uma fusão.

InsereClienteBloqueado: armazena informação a respeito de bloqueio em nodos anteriores.

TestaDesbloqueioEmNoAnterior: caso haja uma fusão bloqueada, verifica se pode efetua-la.

TrataDesbloqueioDeEmergencia: processa o desbloqueio de um ponto de fusão.

RelataEstatistica: apresenta informações coletadas a partir das variáveis de instância.

1. **CLASSE:** PontoDeSincronizacao

2. **DESCRIÇÃO:** representa nodos do tipo ponto de sincronização.

3. **SUPERCLASSES:** NodoVirtual

4. **VARIÁVEIS DE CLASSE:**

(pública):

Vtipo: indica que o objeto é do tipo ponto de sincronização.

5. **VARIÁVEIS DE INSTÂNCIA:**

(pública):

Vquant_filas: indica se o ponto de sincronização possui uma ou duas filas.

Vfila1 e Vfila2: apontadores para objetos da classe Fila.

Ventrada: armazena parâmetros que devem ser seguidos na sincronização.

VnumDeSinconiz: contabiliza as sincronizações que já foram realizadas.

VnumDeBloqueios: contabiliza o número de vezes que o ponto de sincronização esteve no estado bloqueado.

VtempoBloqueado: contabiliza o tempo que o ponto de sincronização ficou no estado bloqueado.

VmomentoDeBloqueio: indica o tempo em que ocorreu o último bloqueio do ponto de sincronização.

VflagDeBloqueio: indica se o ponto de sincronização está no estado bloqueado.

VnumDeClientes: contabiliza o número de clientes no ponto de sincronização.

6. **MÉTODOS**

6.1 - PÚBLICOS:

VisitaNo: processa a visita de um cliente a um ponto de sincronização.

EstadoDisponivel: dada uma rota, indica se o ponto de sincronização está no estado disponível.

Disponibilidade: dada uma rota, retorna o número de vagas na

fila que estiver associada a rota.

SaidaDeEmergência: processa a saída dos cliente associados a uma sincronização.

InsereClienteBloqueado: armazena informação a respeito de bloqueio em nodos anteriores.

TestaDesbloqueioEmNoAnterior: verifica se pode efetuar o desbloqueio de nodo(s) anterior(es).

TrataClienteBloqueadoDeEmergencia: processa o bloqueio de uma sincronização.

TrataDesbloqueioDeEmergencia: processa o desbloqueio de um ponto de sincronização instância. Esta deve proceder a(s) sincronização(ões) que estava(m) bloqueada(s).

RelataEstatistica: apresenta informações coletadas a partir das variáveis de instância.

1. **CLASSE:** PontoEscalonador

2. **DESCRIÇÃO:** representa nodos do tipo ponto escalonador.

3. **SUPERCLASSES:** NodoVirtual

4. **VARIÁVEIS DE CLASSE:**

   (pública):

   Vtipo: indica que o objeto é do tipo ponto escalonador.

5. **VARIÁVEIS DE INSTÂNCIA:**

   (pública):

   Vquant_filas: indica quantas filas o nodo possui.

   VflagDeBloqueio: indica se o ponto escalonador está no estado bloqueado.

   VnumDeClientes: indica quantidade de clientes nas filas do ponto escalonador.

   VnumDeEscalonamentos: contabiliza o número de escalonamentos de clientes que foram feitos.

   VnumDeBloqueios: contabiliza o número de vezes que o ponto escalonador ficou no estado bloqueado.

   VtotCliBloqueados: contabiliza o número de clientes que ficaram bloqueados, isto é, tiveram de esperar nas filas do ponto escalonador.

   VtempoBloqueado: contabiliza o tempo que o ponto escalonador ficou no estado bloqueado.

   VmomentoDeBloqueio: indica o instante em que ocorreu o último bloqueio do ponto escalonador .

   Vfila_rota: apontador para um objeto da classe ListaDe-Filas_Rotas.

   Vescalonador: apontador para um objeto da classe Escalona-dorVirtual. Deve ser utilizado para apontar um objeto de uma das subclasses da classe EscalonadorVirtual.

   VtipoEsc: indica a classe do objeto apontado pela variável Vescalonador.

   VproxNo: indica o proximo nodo de um ponto escalonador.

   VrotaBloq: apontador para um objeto da classe Rota. Serve para indicar ao objeto apontado pela variável VproxNo que ele esta

bloqueando clientes da rota apontada por VrotaBloq.

6. MÉTODOS

6.1 - PÚBLICOS:

PontoEscalonador: construtor.

Inicializa: associa rotas com as filas do ponto escalonador.

VisitaNo: processa a visita de um cliente a um ponto escalonador. Isto é feito através do envio da mensagem VisitaNo para a variável Vfila_rota.

Disponibilidade: dada uma rota, informa a disponibilidade da fila associada.

EstadoDisponivel: dada uma rota, informa se a fila associada está no estado disponível.

Saida: processa a saída de um cliente do ponto escalonador.

InsereClienteBloqueado: armazena informação a respeito de bloqueios em nodos anteriores.

TestaDesbloqueioEmNoAnterior: verifica se há nodo anterior que pode ter cliente(s) desbloqueado(s).

TrataClienteBloqueado: processa cliente que deve continuar no ponto escalonador pois teve seu avanço bloqueado.

TrataDesbloqueioDeEmergencia: processa o desbloqueio de um ponto escalonador.

RelataEstatistica: apresenta informações coletadas a partir das variáveis de instância.

1. **CLASSE:** Relógio

2. **DESCRIÇÃO:** armazena o tempo simulado

3. **SUPERCLASSES:** Não há

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:**

   (pública):

   VtempoCorrente: armazena o tempo simulado.

6. **MÉTODOS**

   6.1 - PÚBLICOS:

   Inicializa: construtor. Ha´dois tipos.

   AtualizaRelogio: atualiza valor de VtempoCorrente.

   ValorDoRelogio: informa valor de VtempoCorrente.

1. **CLASSE:** Rota

2. **DESCRIÇÃO:** armazena a relação de nodos que compoêm uma rota.

3. **SUPERCLASSES:** Não há

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:**

(pública):

Vnome: armazena o nome da rota.

Vclasse: armazena um apontador para um objeto da classe Classe.

Vtipo: identificador de rota fechada ou rota aberta.

Vpopulacao: armazena a quantidade de clientes da rota, caso seja uma rota fechada.

VtotClientes: armazena a quantidade de clientes gerados em uma rota aberta.

VnodosDaRota: apontador (ListaDeNodos) para a lista que armazena nodos.

6. **MÉTODOS**

6.1 - PÚBLICOS:

cria: gera uma instância dessa classe.

DizTipo: informa se rota é aberta ou fechada.

RetornaProxNo: a partir de um nodo informa o nodo que lhe segue na rota.

ProximoNo: informa o conteúdo de RetornaProxNo se o nodo seguinte estiver no estado disponível, senão indica a indisponibilidade.

GuardaRoteamento: armazena uma lista de nodos em VnodosDaRota.

IniciCFechada: inicializa uma rota fechada.

IniciAberta: inicializa uma rota aberta.

PrimeiroNo: informa primeiro nodo de uma rota.

UltimoNo: informa ultimo nodo de uma rota.

PosicionaNClientes: posiciona n clientes num determinado nodo.

ContaSaida: armazena dados a respeito da saída (eliminação) de clientes de uma rota.

RelataEstatisitca: apresenta medidas de desempenho a partir dos dados coletados.

1. **CLASSE:** Sorvedouro

2. **DESCRIÇÃO:** representa nodos do tipo sorvedouro, isto é nodos que eliminam clientes

3. **SUPERCLASSES:** NodoVirtual

4. **VARIÁVEIS DE CLASSE:**

(pública):

Vtipo: indica que o objeto é do tipo sorvedouro.

5. **VARIÁVEIS DE INSTÂNCIA:**

(pública):

VclientesEliminados: contador de clientes que foram eliminados.

Vrota: apontador para um objeto da classe Rota.

6. **MÉTODOS**

6.1 - PÚBLICOS:

Sorvedouro: construtor.

Inicializa: inicializa as variáveis que caracterizam um sorvedouro.

VisitaNo: processa a visita de um cliente a um sorvedouro, isto é procede a eliminação deste cliente.

Disponibilidade: informa que um sorvedouro possui a disponibilidade necessária.

EstadoDisponivel: informa que um sorvedouro está sempre no estado disponível.

RelataEstatistica: apresenta estatísticas obtidas a partir das variáveis de instância.

1. **CLASSE:** VezNoCiclo

2. **DESCRIÇÃO:** armazena o número que identifica uma das filas de um ponto escalonador e sua forma de liberação de clientes.

3. **SUPERCLASSES:** Não há

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:**

(pública):

VnumFila: armazena o número identificador da fila.

VnumLiberacoes: armazena o número de clientes que a fila associada libera, quando esta trabalha na forma não exaustiva.

Vflag: indica se a fila escalonada deve ou não ter seus clientes liberados de forma exaustiva.

6. **MÉTODOS**

6.1 - PÚBLICOS:

InformaNumDaFila: retorna o valor de VnumFila.

InformaNumDeLiberacoes: retorna o valor de VnumLibercaoes.

InformaFlag: retorna o valor de VFlag.

1. **CLASSE:** zDLCursor

2. **DESCRIÇÃO:** serve para implementação de listas duplamente encadeadas. Esta classe pertence à biblioteca de classes que acompanha o compilador Zortech C++.

3. **SUPERCLASSES:** Não há

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTANCIA:**

(pública):

current: apontador para elemento da lista.

root: apontador para o começo da lista.

6. **MÉTODOS**

6.1 - PÚBLICOS:

fwd: atualiza current com o endereço do próximo elemento da lista.

bkwd: atualiza current com o endereço do elemento anterior na lista.

start: atualiza current com o endereço do primeiro elemento da lista.

end: atualiza current com o endereço do último elemento da lista.

get: retorna o elemento da lista apontado por current.

update: atualiza o conteúdo do elemento apontado por current.

1. CLASSE: zDList

2. DESCRIÇÃO: classe base das listas utilizadas. E duplamente encadeada.

3. SUPERCLASSES: Não há

4. VARIÁVEIS DE CLASSE: Não há

5. VARIÁVEIS DE INSTANCIA:

(pública):

current: apontador para elemento da lista.

root: apontador para o começo da lista.

6. MÉTODOS

6.1 - PÚBLICOS:

fwd: atualiza current com o endereço do próximo elemento da lista.

bkwd: atualiza current com o endereço do elemento anterior na lista.

start: atualiza current com o endereço do primeiro elemento da lista.

end: atualiza current com o endereço do último elemento da lista.

get: retorna o elemento da lista apontado por current.

update: atualiza o conteúdo do elemento apontado por current.

1. **CLASSE:** zDLnode

2. **DESCRIÇÃO:** elemento armazenado em zDList

3. **SUPERCLASSES:** Não há

4. **VARIÁVEIS DE CLASSE:** Não há

5. **VARIÁVEIS DE INSTÂNCIA:**

   (pública):

   next: apontador para o próximo elemento da lista.

   prev: apontador para o próximo elemento da lista.

   body: apontador para o conteúdo de um elemento da lista.

6. **MÉTODOS:** Não há

REFERENCIAS

BIBLIOGRAFICAS

REFERÊNCIAS BIBLIOGRÁFICAS

[Brasileiro 89] Brasileiro, Marcos Antonio Gonçalves, Cabral, Maria Izabel C. & Silva, Hélio Menezes. **SAVAD - Uma Ferramenta para Avaliar o Desempenho de Sistemas Distribuídos.** Seminário Franco-Brasileiro de Sistemas Distribuídos. Florianópolis, 1989, pp 281-288.

[Cabral 87] Cabral, Maria Izabel C. **Modelagem do Protocolo da Camada de Sessão.** Tese de Doutorado. CCPgEE/CCT/UFPB. 1987.

[Cabral 90] Cabral, Maria Izabel C. **Curso de Avaliação de Desempenho em Redes de Computadores**, I JONNAI - UFPB/IBM, Campina Grande, 1990

[Cabral 92] Cabral, Maria Izabel Cavalcanti, Silva, H. M. & Brasileiro, M.A. **The Object-Oriented Design of an Intelligent Interface for Modelling Networks of Queues**, XII International Conference of Chilean Computer Science Society, Santiago, Chile, 1992

[Cabral 93a] Cabral, Maria Izabel Cavalcanti **Ciclo de Vida de Simuladores Discretos Orientados a Objetos.** Tese para Concurso para Professor Titular. DSC/CCT/UFPB. 1993.

[Cabral 93b] Cabral, Maria Izabel C, Filho, Haroldo Castro C, Brasileiro, Marcos A.G., & Silva, Hélio Menezes. **A Simulator for Performance Evaluation of Models for Networks of Queues**, 1993 European Simulation Multiconference - ESM93, Lyon, França, 1993

[Cauper 86] Cauper, Antonio. **Um simulador para avaliação de desempenho de redes locais.** Dissertação de Mestrado. UFPB, 1988.

[Coimbra 91] Coimbra, Laura - **Utilização de Orientação para Objetos** - Mundo UNIX, número 26, Computerworld do Brasil, maio de 1991

[Dias 92] Dias, Maria Madalena. **SIMILE - Um Simulador Reutilizável para Avaliação de Desempenho de Redes Locais.** Dissertação de Mestrado. UFPB. 1992.

[Dewhurst 90] Dewhurst, Stephen C. & Stark, Kathy T. **Programando em C++**, Editora Campus, 1990.

[Dietsch 91] Diesch, Kurt H. & Barta, Thomas A. **Object-oriented discrete-event simulation in a strongly-typed procedural language.** Simulation Series. Vol. 23, num 3, 1991, pp. 43-49.

[Doyle 90] Doyle, Robert J. **Object-oriented simulation programming.** Simulation Series. 1990, pp. 1-6.

[Goldberg 83] Goldberg, A. & David, R. **Smalltalk-80: The Language and its Application.** Addison-Wesley, Reading, MA. 1983

[Kernighan 78] Kernighan, B.W. & Ritchie, D.M. **The C Programming Language.** Prentice-Hall. 1978.

[Kleinrock 75] Kleinrock, L. **Queueing Systems, vol 1: Theory,** Wiley Interscience, N.Y., 1975

[Kreutzer 86] Kreutzer, Wolfgang **System Simulation: Programing styles and languages**, Addison-Wesley, London, 1986

[Moura 86] Moura, Jose Antão Beltrão et alii. **Redes Locais de Computadores - Protocolos de Alto Nível e Avaliação de Desempenho.** Editora McGraw-Hill. 1986.

[Pinson 91] Pinson, Lewis J. & Wiener, Richard S. **C++: programação orientada para objeto: manual prático e profissional.** Editora Makron Books, SP, 1991

[Rentsch 92] Rentsch, T. **Object-Oriented Programming.** SIGPLAN Notices, Vol. 17, Número 9, Setembro de 1992, pp. 76-87

[Sauer 81] Sauer, C.H. & Chandy, K.M. **Computer Systems Performance Modeling,** Prentice-Hall, Inc., Englewood Cliffs, N.J., 1981,.

[Shannon 86] Shannon, R. I. **Intelligent Simulation Environment.** Simulation Series. Vol. 17., num. 1, 1986, pp. 150-156.

[Soares 90] Soares, Luis Fernando G. **Modelagem e Simulação Discreta de Sistemas**, VII Escola de Computação, 1990.

[Souto 92] Souto, Francisco A. Coutinho, Cabral, M.I.C, Brasileiro, M.A.G. & Silva,H.M. **Projeto Orientado a Objetos de um Sistema Especialista para Avaliação de Desempenho de Modelos de Redes de Filas**, XII Congresso da SBC, Rio de Janeiro, outubro de 1992.

[Takahashi 88] Takahashi, Tadao. **Introdução a Programação Orientada a Objetos.** III EBAI, 1988.

[Takahashi 90] Takahashi, Tadao. **Programação Orientada a Objetos.** VII Escola de Computação, 1990.

[Zortech 90] Zortech C++ Compiler V2.1. C++ Tools. 1990.