UNIVERSIDADE FEDERAL DA PARAIBA

CENTRO DE CIENCIAS E TECNOLOGIA

COORDENAÇÃO DE POS-GRADUAÇÃO EM INFORMATICA

# SIMILE - Um Simulador Reutilizável para Avaliação de Desempenho de Redes Locais

MARIA MADALENA DIAS

Campina Grande

ABRIL/1992

MARIA MADALENA DIAS

# SIMILE - Um Simulador Reutilizável para Avaliação de Desempenho de Redes Locais

Dissertação apresentada ao Curso de MESTRADO EM INFORMATICA da Universidade Federal da Paraíba, em cumprimento às exigências para obtenção do Grau de Mestre.

AREA DE CONCENTRAÇÃO: CIENCIA DA COMPUTAÇÃO

Maria Izabel Cavalcanti Cabral

Orientadora

Marcos Antonio Gonçalves Brasileiro

Co-Orientador

Campina Grande

ABRIL/1992


D541s Dias, Maria Madalena
SIMILE : um simulador reutilizavel para avaliacao de desempenho de redes locais / Maria Madalena Dias. - Campina Grande, 1992.
140 f. : il.

Dissertacao (Mestrado em Informatica) - Universidade Federal da Paraiba, Centro de Ciencias e Tecnologia.

1. Redes Locais 2. Simile 3. Ciencia da Computacao 4. Informatica 5. Dissertacao I. Cabral, Maria Izabel Cavalcanti, Dra. II. Brasileiro, Marcos Antonio Goncalves, Dr. III. Universidade Federal da Paraiba - Campina Grande (PB) IV. Título

CDU 004.732(043)

SIMILE - Um Simulador Reutilizável para Avaliação de Desempenho de Redes Locais

**MARIA MADALENA DIAS**

DISSERTAÇÃO APROVADA EM 13/04/1992

MARIA IZABEL CAVALCANTI CABRAL - Dr.
ORIENTADORA

MARCOS ANTONIO GONÇALVES BRASILEIRO - PhD.
CO-ORIENTADOR

JOSÉ ANTÃO BELTRÃO MOURA - PhD.
COMPONENTE DA BANCA

EDMUNDO ALBUQUERQUE DE SOUZA E SILVA - PhD.
COMPONENTE DA BANCA

Campina Grande, 13 de Abril de 1992

Aos meus pais

## AGRADECIMENTOS

Agradeço aos meus orientadores, Profa.Dra. Maria Izabel Cavalcanti Cabral e Prof.Dr. Marcos Antonio Gonçalves Brasileiro, pelo apoio e dedicação a mim dispensados.

Agradeço ao Prof.Dr. José Antão Beltrão Moura pela colaboração na validação do SIMILE.

Agradeço, também, aos meus colegas do Curso de Mestrado pelo apoio oferecido no decorrer desse curso, especialmente ao Francisco Araripe.

Agradeço aos meus pais que, apesar de virem de famílias humildes e terem pouco estudo, sempre acreditaram que a maior riqueza que os pais podem dar aos seus filhos é o estudo.

# SUMARIO

## LISTA DE FIGURAS

# LISTA DE TABELAS

## RESUMO

Com a crescente demanda de utilização de Redes Locais de Computadores, tem havido uma maior preocupação em avaliar o desempenho dessas redes. Esta dissertação apresenta um Simulador Reutilizável para Avaliação de Desempenho de Redes Locais, denominado SIMILE. Esse simulador projetado para Redes Locais de Computadores com Protocolos de Acesso ao Meio conforme Padrão IEEE 802, objetiva, também, servir como protótipo de simulação de Redes Locais de Computadores que possa ser facilmente expandido para o estudo e a avaliação de outros protocolos de comunicação. Essa facilidade deve-se às características de modularidade, extensibilidade e reutilização de software propiciadas pela Abordagem Orientada a Objetos utilizada no desenvolvimento do SIMILE.

## Abstract

The potencial use of Local Area Networks-LANs in many applications has made, presently, the performance evaluation of those LANs an important research topic. This thesis addresses a reusable simulator named SIMILE, developed for performance evaluation of LANs, in particular those LANs in the IEEE 802 standard. The Object-Oriented Approach is considered in the implementation of SIMILE. As a result, characteristics such as modularity, extensibility and reusability are inherent to this reusable simulator. These characteristics make SIMILE a versatile performance evaluation simulator that can be easily adapted for performance evaluation of LANs and protocols other than those specified in the IEEE 802 standard.

# C A P I T U L O I

## INTRODUÇÃO

A avaliação da habilidade de um sistema em suportar a demanda de sua utilização é essencial nas fases de projeto e operação. Essa demanda é caracterizada pelo tipo de aplicação (tráfego) na entrada do sistema.

Na avaliação de desempenho de um sistema, define-se um conjunto de medidas de desempenho de interesse (relevantes), e observa-se o comportamento destas medidas face às variações na demanda da sua utilização.

A avaliação de desempenho de Redes Locais de Computadores (RLCs) é feita tipicamente por três técnicas que obtêm relações quantitativas entre as medidas de interesse: Simulação Digital, Tratamento Analítico (Teoria dos Processos Estocásticos/Teoria das Filas) e Medição de Desempenho (obtenção de tempos de resposta, vazão, etc.).

Nas técnicas Simulação Digital e Tratamento Analítico, usadas na avaliação de desempenho de RLCs, se faz necessária a construção de modelos para a observação do comportamento das medidas de desempenho relevantes. A modelagem de um sistema, além de possibilitar o estudo do

seu comportamento de forma controlada, pode permitir uma compreensão profunda do seu funcionamento [6].

Do ponto de vista de simulação, um modelo pode ser construído para destacar aspectos mais importantes do problema e ignorar os menos relevantes [20].

As mudanças, isto é, as alterações nas variáveis de estado do sistema, são a base para a classificação dos modelos. Os modelos de um sistema podem ser classificados em modelos de mudança discreta e modelos de mudança contínua [27]. O modelo de mudança discreta, ou simplesmente modelo discreto, é aquele em que as trocas de estado são instantâneas e ocorrem em pontos discretos do tempo. O modelo de mudança continua, ou simplesmente modelo contínuo, é aquele em que as trocas de estado ocorrem continuamente no tempo. O modelo de mudança combinada, ou simplesmente modelo combinado, é aquele em que as trocas de estado podem ocorrer discretamente, continuamente, ou continuamente com saltos discretos superpostos.

Redes Locais de Computadores são sistemas que apresentam contenção de recursos. Desta forma, modelos de redes de filas se adequam à modelagem destes sistemas. Esses modelos são classificados como modelos de mudança discreta.

Num sistema de filas, os fregueses, em geral, esperam em fila para serem atendidos e, após receberem o serviço desejado do(s) servidor(es) no sistema, podem permanecer no sistema para receberem novo serviço ou partir do sistema.

Informalmente, um modelo de redes de filas consiste de um conjunto de estaçoes (ou nodos) de serviço conectados de forma a atender os requerimentos de serviços de fregueses. Cada estação de serviço possui um ou mais servidores e uma ou mais filas onde fregueses permanecem enquanto esperam para serem servidos.

As redes de filas são classificadas em fechadas, abertas e mistas. Define-se como rede fechada de filas uma rede de filas consistindo de $M$ nodos interligados, numerados $1,2,...,M$, com uma topologia especifica da rede. Não há chegadas externas de fregueses e nem partidas para o exterior, assim a população é fixa e finita. Com rede aberta de filas, a população varia estatisticamente. As redes mistas englobam características das redes fechadas e abertas [6].

Um modelo de redes de filas pode ser solucionado analiticamente ou numericamente. A solução analitica é obtida através da resolução de equações matemáticas que relacionam os parâmetros do modelo com as medidas de desempenho de interesse. A Teoria das Filas é uma das técnicas utilizadas na solução analitica. A solução numérica pode ser obtida através de interações, métodos de convergëncia ou interpolação.

Apesar da solução analitica ser mais econömica e eficiente, muitas vezes sua aplicação é limitada pela complexidade do sistema que está sendo modelado. Para esse

tipo de situação, na maioria das vezes, a única solução possível é a Simulação Digital.

## 1.1 Simulação Digital

A Simulação Digital é uma técnica usada para prever o comportamento de sistemas reais ou hipotéticos. Ela utiliza um modelo para obter relações entre as medidas de desempenho de interesse.

A Simulação Digital permite experimentação automatizada com sistemas que não foram concebidos ou que são difíceis de serem manuseados. Ela também permite a experimentação repetida com um sistema, sob condições controladas, para otimizar seu desempenho.

A simulação pode ser discreta, contínua ou combinada, conforme o modelo utilizado (discreto, contínuo ou combinado, respectivamente).

Os objetos em um sistema discreto são chamados entidades. Existem vários tipos de entidades e cada um tem vários tipos de características ou atributos.

O objetivo da simulação discreta é reproduzir as atividades das entidades envolvidas e, a partir daí, identificar aspectos relevantes sobre o comportamento e desempenho do sistema. Isto é conseguido através da definição dos estados do sistema e da construção das atividades que mudam o estado do sistema.

O estado de um sistema é definido em termos de valores numéricos dos atributos das entidades. Em simulação

discreta, o estado do sistema só pode mudar nos tempos de ocorrência de eventos [27].

A simulação discreta pode ser orientada a evento, ao exame da atividade ou a processo. Na simulação orientada a evento, um sistema é modelado pela definição das mudanças que ocorrem no tempo do evento. A simulação do sistema é produzida pela execução da lógica associada a cada evento, em uma sequência ordenada no tempo [27].

Um evento é uma perturbação instantânea que muda o estado do sistema (e por conseguinte, do modelo). Um modelo de simulação acionada por eventos deve: identificar os eventos que podem ocorrer no sistema; avaliar os efeitos dos eventos no estado e atividade do sistema; permitir a ocorrência dos eventos e atualizar o estado e atividades do sistema à medida que os eventos ocorrem [23].

Para que os eventos possam ser identificados no momento de suas ocorrências, é introduzido no modelo de simulação uma variável chamada "relógio simulado" para representar o "tempo simulado". Em um simulador acionado por eventos, o "relógio simulado" avança para o instante de "tempo simulado" em que ocorrerá o próximo evento.

Na simulação orientada ao exame da atividade, são descritas as atividades nas quais as entidades do sistema estão engajadas e são relacionadas as condições que causam o início e o fim de uma atividade. Os eventos que iniciam ou terminam uma atividade não são escalonados, mas são iniciados a partir das condições especificadas para a

atividade. A medida que o tempo simulado avança, as condições para o início ou o fim de uma atividade são examinadas. Na simulação orientada a processo, é especificada a lógica das seqüências de eventos no modelo, as quais ocorrem em padrões definidos, para modelar o fluxo das entidades no sistema [27].

## 1.2 Abordagem Orientada a Objetos

A programação orientada a objetos oferece uma alternativa para o ambiente tradicional de simulação e suas linguagens. Na verdade, a essência da programação orientada a objetos é a simulação; ela fornece uma forma mais natural de abstrair o modelo do que é possível nas linguagens de programação convencionais, ou mesmo nas linguagens de simulação atuais [1].

A programação orientada a objetos tem provado ser um estilo de programação muito elegante e produtivo para lidar com modelos e sistemas de software complexos [17].

Com o surgimento da programação orientada a objetos, sentiu-se a necessidade de desenvolver o método de projeto orientado a objetos pois, segundo Booch em [3], apesar da utilização do método de projeto estruturado no projeto de sistemas convencionais dar ótimos resultados, ele não se adequa muito bem para sistemas extremamente complexos e é inapropriado para o projeto de sistemas que utilizam liguagens de programação orientadas a objetos.

O projeto orientado a objetos, como outras metodologias de projeto orientado a informação, cria uma representação do domínio do problema do mundo real e mapeia este problema em um domínio de solução que é o software [25].

O projeto de um sistema de software orientado a objetos é construído em torno de um conjunto de classes que caracterizam o comportamento dos objetos do mundo real relativos ao sistema. Em geral, uma classe é a descrição de um conjunto de objetos que possuem os mesmos atributos e comportamentos e, um objeto é uma instância específica de uma classe com comportamento bem definido. Os objetos de cada classe são manipulados através do envio de mensagens. Estas mensagens representam o conjunto de ações que são tomadas sobre o conjunto de objetos.

Na simulação, os objetos correspondem aos objetos do mundo externo, e representam componentes (entidades) modulares do mundo real. O comportamento dos objetos do modelo de simulação descreve o comportamento dos objetos externos, ou seja, como estes objetos irão se comportar em resposta a várias entradas (eventos). Objetos agem sobre outros objetos através da passagem de mensagens, descrevendo ambas as ações funcional e relacional. Mensagens passadas entre objetos são as responsáveis por todas as suas interações [1].

Mensagens podem recuperar ou modificar valores de variáveis privadas ou disparar uma operação (leitura,

impressão, movimento do cursor na tela, etc.). A comunicação entre os objetos só pode ser feita através de mensagens [17].

Programas orientados a objetos são construídos de componentes de software reutilizáveis. Basta que uma classe seja implementada e testada para que ela possa ser usada como um bloco na construção de um programa. Se novas características ou alterações são requeridas, novas classes podem ser derivadas das classes já existentes. Sistemas tornam-se mais fáceis de ser testados considerando que os erros de programas podem ser isolados nos novos códigos das classes derivadas [19].

Como um conjunto de classes é criado, torna-se mais fácil desenvolver novas aplicações que utilizam tais classes. Programadores que trabalham em grupo podem construir uma aplicação como uma série de classes, que são posteriormente organizadas para formar um programa final [19].

Um dos maiores objetivos da programação orientada a objetos é aumentar a produtividade do programador através das facilidades de extensibilidade e reutilização de software propriciados por ela [10].

Linguagens de programação orientadas a objetos oferecem muitas vantagens sobre linguagens de programação tradicionais que facilitam a construção e a manutenção de programas de simulação. Tais vantagens são: encapsulamento

(*information hiding*), abstração de dados, herança e amarração dinâmica (*dynamic binding*) [10].

Encapsulamento refere-se à quebra do programa em módulos que possam ser modificados de forma independente. Todo módulo é um objeto, isto é, uma estrutura de dados que contém os procedimentos que operam sobre ela. Cada objeto contém seus próprios dados e esconde esses dados dos outros objetos.

Abstração de dados é o processo de esconder estruturas de dados dentro de objetos. Esta prática evita os fortes requisitos de checagem de tipos de muitas linguagens de programação. Procedimentos dentro do objeto agem sobre os dados independente do tipo. Estes procedimentos são chamados "métodos". O relacionamento direto entre dados e procedimentos tornam os programas orientados a objetos mais fáceis de ser mantidos.

Linguagens orientadas a objetos compartilham códigos através do mecanismo de herança. Uma nova classe de objetos pode ser criada como uma variação ou cópia exata de uma classe existente. A nova classe é chamada uma subclasse da classe antiga, e a classe antiga é uma superclasse da nova classe. Objetos na subclasse herdam todas ou quase todas as propriedades da superclasse, incluindo a implementação de métodos. A subclasse pode definir métodos adicionais e redefinir métodos antigos. É através do mecanismo de herança que hierarquias de classes são construídas.

A amarração dinâmica ocorre na execução do programa, quando um método da classe derivada é redefinido por possuir funcionamento diferente do método da classe base correspondente. Somente em tempo de execução é determinado qual dos objetos receberá a mensagem. Isto é conhecido como polimorfismo. O resultado final é que um único nome de método pode ser usado para a mesma operação desempenhada pelas classes derivadas, mesmo se a implementação daquele método varie de classe a classe.

Amarração dinâmica adicionada ao mecanismo de herança facilitam a extensibilidade e a reutilização do software. A extensibilidade do software é obtida através da inclusão de novas classes de objetos. A reutilização do software é obtida através da utilização de classes de objetos já implementadas, ou através da criação de novas classes derivadas das classes já existentes.

Entre as linguagens de programação orientadas a objetos mais conhecidas destaca-se a linguagem C++, que foi desenvolvida por Bjorne Stroustrup nos Laboratórios Bell no inicio dos anos 80 para suportar a escrita de alguns simuladores complexos [31].

## 1.3 Objetivos

Esta dissertação objetiva a construção de um Simulador Reutilizável para Avaliação de Desempenho de Redes Locais, denominado SIMILE. O projeto e desenvolvimento desse simulador foram baseados no método de projeto orientado a

objetos e na programação orientada a objetos. Sua implementação foi feita na linguagem de programação C++.

O SIMILE é um simulador discreto orientado a eventos. Ele foi projetado para RLCs com Protocolos de Acesso ao Meio conforme Padrão IEEE 802 (Protocolos Passagem de Ficha em Anel e em Barra e Barra com CSMA-CD) e objetiva, também, servir como protótipo de simulação de RLCs que pode ser facilmente expandido para o estudo e a avaliação de outros protocolos de comunicação. Essa facilidade deve-se às características de modularidade, extensibilidade e reutilização de software propiciadas pela abordagem orientada a objetos.

O SIMILE oferece, também, aos seus usuários as seguintes facilidades:

1) Uma interface amigável que habilita seus usuários a facilmente: a) escolher e especificar o seu modelo e as medidas de desempenho desejadas e, b) receber os resultados da simulação solicitados.

2) O fornecimento das medidas de desempenho relevantes para cada modelo simulado (exemplos: vazão média da sub-rede de comunicação, tempo médio de transmissão de um pacote na sub-rede de comunicação, tempo médio de espera de um pacote em fila, atraso médio fim-a-fim e a taxa de utilização do meio de transmissão). O SIMILE fornece, também, medidas pertinentes às aplicações específicas consideradas no modelo (como por exemplo, em aplicações

criticas no tempo [5], a cauda da distribuição do atraso fim-a-fim.

3) Intervalos de confiança obtidos automaticamente através da análise dos resultados de um número *n* de execuções do simulador.

4) Transparência da Aplicação. O usuário define o tráfego de entrada do seu modelo conforme a sua aplicação. A definição desse tráfego poderá ser feita das seguintes formas: a) a partir de uma expressão analítica (processo estocástico), e b) através da leitura de um arquivo com dados acerca desse processo estocástico. O usuário também define o número de interfaces por classe[1] de tráfego e as medidas de desempenho de interesse para cada aplicação considerada.

5) Possibilidade de ampliar o SIMILE adicionando aos modelos do Protocolo de Acesso ao Meio do Padrão IEEE 802 aqui proposto, outras opções de modelos. Esta ampliação é facilitada pelas caracteristicas de modularidade, extensibilidade e reutilização de software, propiciadas pela abordagem orientada a objetos e utilizadas no desenvolvimento do SIMILE.

6) Utilização de multiclasses. Esta facilidade permite a avaliação de desempenho de redes integradas, como por exemplo, a integração de voz e dados. Esse tipo de

---

1. O termo classe, aqui empregado, diz respeito aos tipos de pacotes (voz, dados, imagens, etc.) que estão especificados nos modelos de RLCs apresentados no próximo capítulo. O termo classe de objetos, usado anteriormente, representa um conjunto de objetos com as mesmas características.

integração é frequentemente utilizado em aplicações de automação de escritório.

O SIMILE é, também, parte integrante do módulo solução do projeto denominado "SAVAD - Uma Ferramenta Inteligente para Avaliação de Desempenho de Sistemas Distribuídos" (projeto de pesquisa multidisciplinar das áreas de Redes de Computadores e Inteligência Artificial da UFPb, financiado pelo CNPq [6]).

## 1.4 Contribuição Científica

Para avaliar o desempenho de RLCs, normalmente, usam-se Linguagens de Simulação de Propósito Geral, tais como GPSS [26], SIMULA [2], RESQ [21] e SIMSCRIPT [7], ou projetam-se simuladores para aplicações específicas dessas redes, como por exemplo [23] e [8]. O SIMILE apresenta-se como uma opção mais abrangente desta última, constituindo-se de um simulador reutilizável para avaliar o desempenho de RLCs que, facilmente, poderá ser estendido ou modificado por usuários projetistas de RLCs, para atender a outras aplicações ou outros requisitos além daqueles oferecidos na sua versão inicial. As características de extensibilidade e reutilização de software surgem naturalmente no projeto do SIMILE, devido à abordagem orientada a objetos utilizada no seu desenvolvimento.

O SIMILE é uma ferramenta amigável onde o usuário, necessariamente, não precisa ter conhecimentos profundos em modelagem e avaliação de desempenho de RLCs.

No SIMILE é permitida a utilização de multiclasses, facilitando assim, a avaliação de desempenho de redes integradas (como por exemplo a integração de voz e dados).

O SIMILE contribuirá, também, no desenvolvimento científico do Projeto Orientado a Objetos e da Programação Orientada a Objetos, especialmente aplicados ao projeto e a implementação de programas simuladores.

Outra contribuição importante está relacionada a sua integração ao módulo solução do projeto "SAVAD - Uma Ferramenta Inteligente para Avaliação de Desempenho de Sistemas Distribuídos", já mencionado neste capítulo.

## 1.5 Organização

O restante desta dissertação está organizado em cinco capítulos e dois apêndices. No capítulo II são descritos os modelos utilizados pelo SIMILE. No capítulo III são relacionados os principais elementos de simulação considerados no desenvolvimento do SIMILE e, também, é mostrada a sua interface. No capítulo IV é apresentado o projeto orientado a objetos do SIMILE. No capítulo V são considerados estudos de casos e analisados seus resultados. No capítulo VI apresentam-se as conclusões e sugestões de continuação desta linha de pesquisa. No apêndice A são apresentadas as fórmulas de cálculo das Funções de Distribuição de Probabilidade (FDP) usadas no SIMILE. No apêndice B são descritos os protocolos das classes de objetos do SIMILE.

O usuário interessado apenas na utilização do SIMILE pode consultar apenas os capítulos III e V. O usuário que queira conhecer detalhes sobre os modelos utilizados no desenvolvimento do SIMILE, deve ler também o capitulo II. Já para aquele usuário que deseja conhecer mais profundamente como o SIMILE foi projetado e implementado, aconselha-se a leitura dos demais capítulos.

# C A P I T U L O II

## MODELOS DE REDES LOCAIS DE COMPUTADORES

O modelo é uma abstração do sistema em estudo. Ele deve refletir o sistema em detalhes suficientes para que a observação do seu comportamento possibilite prever ou estimar o comportamento do sistema. A modelagem de um sistema pode ser vista como uma "simplificação" que descarta as caracteristicas julgadas irrelevantes do sistema [23].

A modelagem requer conhecimento completo do sistema, intuição e habilidade.

Ao modelar um sistema, deve-se procurar abstrair os elementos considerados importantes ao sistema e as relações entre esses elementos que são pertinentes à resolução do problema. Passa-se assim, por dois graus de refinamento, primeiro na definição do sistema e, depois, na definição do modelo [27].

As etapas na construção de um modelo são:

a) Estabelecer a estrutura do modelo : determinar os limites do sistema (com o seu ambiente), identificar as entidades, os atributos e as atividades deste sistema.

b) Fornecer valores aos atributos e definir as interações existentes entre suas entidades.

Neste capítulo é feita uma descrição dos modelos para o estudo de protocolos de acesso, Padrão IEEE 802, utilizados no desenvolvimento do SIMILE. Maiores detalhes sobre esses modelos poderão ser encontrados em [5, 15, 23, 28, 30]. Também neste capítulo, é apresentado o modelo do servidor único e são relacionadas as características gerais desse modelo.

## 2.1 Descrição dos Modelos

Seguem a descrição dos modelos de RLCs utilizados no estudo de protocolos de acesso, conforme Padrão IEEE 802 [15].

### 2.1.1 Modelo de RLC em Anel com Passagem de Ficha

Um anel com ficha consiste de um conjunto de interfaces conectadas serialmente por um meio de transmissão. A informação é transferida sequencialmente de interface a interface.

Uma ficha é um sinal de controle compreendido de uma seqüência única de sinalização que circula no meio, seguindo cada transferência de informação.

A figura 2.1 apresenta o modelo de RLC em Anel com Passagem de Ficha.

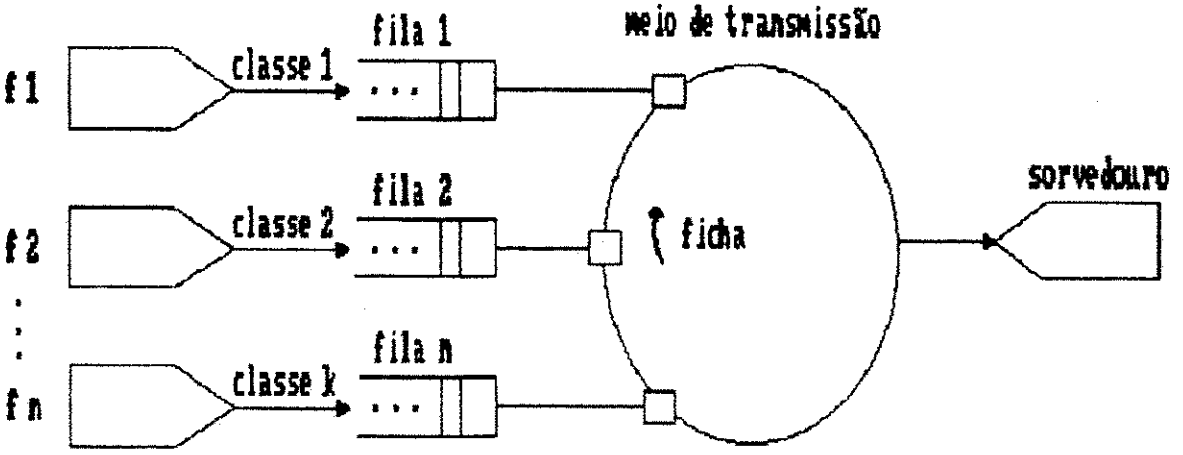


**Fig. 2.1 - Rede Local em Anel com Passagem de Ficha**

Nessa figura, *fi* (*i*=1,2,...,n) representa a fonte de geração de pacotes em cada interface *i*. Cada interface possui um tipo de classe (k, k<=n) e uma fila com armazenamento de pacotes, cujo comprimento (L) pode ser limitado ou ilimitado.

De acordo com o Protocolo de Comunicação para RLC com Passagem de Ficha, quando uma interface *i* recebe uma ficha livre, esta adquire o direito de transmitir um ou mais pacotes, conforme uma disciplina de liberação de pacotes. Após a transmissão do(s) pacote(s), a interface *i* emite uma ficha livre que é passada para a interface seguinte na ordem física.

Quando um pacote chega a sua respectiva fila e esta encontra-se vazia, ele deve esperar até que a ficha livre chegue à interface para a sua transmissão; caso contrário,

além de esperar pela visita da ficha livre, ele deve esperar, também, que todos os pacotes que estiverem a sua frente sejam transmitidos (conforme disciplina de escalonamento). Neste caso, o tempo de serviço para um pacote que esteja na iminência de ser servido corresponde ao tempo de transmissão do pacote, mais o tempo de espera da visita da ficha. Este último tempo é uma variável aleatória que depende das características dos tráfegos presentes, do comprimento da ficha, do meio de transmissão, e do número de interfaces ativas (com pacotes a transmitir) durante um ciclo da ficha [5]. O ciclo da ficha corresponde ao tempo gasto na passagem da ficha de uma interface à interface seguinte.

De acordo com as considerações acima, uma interface pode ser modelada conforme a figura 2.2. Nota-se nessa figura que a visita da ficha livre equivale ao fechamento de uma porta de controle, para a transmissão de pacote(s) na sub-rede de comunicação [5]. Nesse modelo, o tempo de chaveamento é ignorado. Este sistema de fila corresponde, em geral, a uma fila G/G/1, cuja solução apresenta-se em [16]. A solução analítica desta fila poderá ser extremamente difícil, dependendo da natureza dos processos de chegada e de serviço de fregueses.


fig. 2.2 - Modelo da Interface

O mecanismo de prioridade do modelo, proposto no Padrão IEEE 802.5 [15], fornece quatro níveis de serviço que correspondem à prioridade de acesso ao meio por um pacote. Estes níveis são identificados como 0, 2, 4 e 6, sendo a classe 6 a de mais alta prioridade e a classe 0 a de mais baixa prioridade. Qualquer interface que não utiliza o parâmetro de prioridade opcional, transmitirá todos os pacotes de dados com uma classe igual a 6.

Neste mecanismo todo pacote transmitido (inclusive a ficha livre) tem um campo designando à prioridade corrente (P) e um outro para a reserva de prioridade (R). Uma interface só pode transmitir um pacote se receber a ficha livre e se o pacote tiver prioridade maior ou igual à designada no campo P da ficha livre.

Quando uma interface recebe a ficha livre, ela transmite os pacotes com prioridade maior ou igual à prioridade passada na ficha livre, até que estes pacotes sejam todos transmitidos (no caso da disciplina de liberação de pacotes ser exaustiva), ou o limite de número de pacotes transmitidos seja atingido (no caso da disciplina de liberação de pacotes ser limitada) ou seu tempo limite de transmissão de pacotes se esgote (no caso da disciplina de liberação de pacotes ser exaustiva ou limitada). Quando a disciplina de liberação de pacotes é não-exaustiva, a interface transmite o pacote de maior prioridade, se este tem prioridade maior que a da ficha livre.

Se a interface não possui nenhum pacote com prioridade (Pp) maior que a prioridade da ficha livre (P) e nem a reserva de prioridade (R) é maior que a da ficha livre, então a nova ficha livre é transmitida com a mesma prioridade da ficha anterior e o campo de reserva com o maior valor entre Pp ou R anterior.

Se no entanto, a interface possuir pacote com prioridade (Pp) maior do que a prioridade da ficha livre anterior (P), ou então esta última for menor que a reserva de prioridade (R), a nova ficha livre será gerada com sua prioridade tendo o maior valor entre Pp e R, e seu campo de reserva igual a 0 (zero).

Todo pacote, após ser transmitido pelo meio de transmissão, entra no nodo sorvedouro, onde é eliminado do

modelo, significando que esse pacote foi transmitido pela RLC e entregue ao seu respectivo destino.

### 2.1.2 Modelo de RLC em Barra com Passagem de Ficha

A figura 2.3 apresenta o modelo de RLC em Barra com Passagem de Ficha.

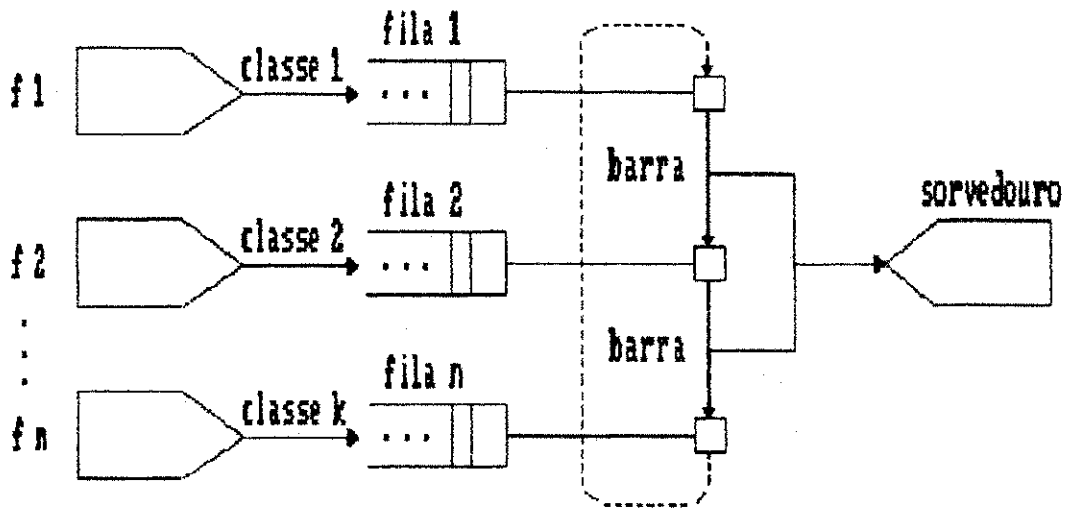


**Fig. 2.3 - Rede Local em Barra com Passagem de Ficha**

Nessa figura, *fi* (*i*=1,2,...,n) representa a fonte de geração de pacotes em cada interface *i*. Cada interface possui um tipo de classe (k, k<=n) e uma fila de pacotes, cujo comprimento (L) pode ser limitado ou ilimitado.

Cada interface de uma RLC em Barra com Passagem de Ficha pode ser modelada por um servidor único com porta de controle (figura 2.2), o barramento físico (meio de

transmissão). Nele as interfaces são conectadas e suas informações são transferidas sequencialmente.

O funcionamento de uma RLC em Barra com Passagem de Ficha é similar ao funcionamento de uma RLC em Anel com Passagem de Ficha, onde existe uma ficha livre que é passada entre as interfaces ativas da rede. Essa ficha livre é responsável pelo controle do meio físico. Quando uma interface detém a ficha, ela tem o controle sobre o meio de transmissão.

A principal diferença entre a RLC em Barra com Passagem de Ficha e a RLC em Anel com Passagem de Ficha decorre do fato de que a barra não impõe uma ordenação natural (física) entre as interfaces tal como no caso do anel, por este motivo torna-se necessária a criação de um anel lógico ou virtual sobre a estrutura de barra que possibilite a operação do protocolo. Cada interface deve manter pelo menos os endereços do seu sucessor e antecessor para que seja preservada a integridade do anel lógico [29].

É importante observar que a ordem física de conexão nada tem a ver com a ordem lógica no anel e que mesmo interfaces que não pertencem ao anel virtual podem receber mensagens, embora não possam transmitir [27].

A principal vantagem da RLC em Barra com Passagem de Ficha em relação à RLC em Anel com Passagem de Ficha é a confiabilidade, por não haver elementos ativos no meio físico. Por outro lado, as interfaces da RLC em Barra com Passagem de Ficha não tem capacidade de alterar bits durante

a transmissão de um pacote, o que tem como principal conseqüência uma menor eficiência do protocolo [29].

A desvantagem da RLC em Barra com Passagem de Ficha é sua complexidade. Uma outra desvantagem é o *overhead* envolvido quando o tráfego é baixo [28].

### 2.1.3 Modelo de RLC em Barra com CSMA-CD

A figura 2.4 apresenta o modelo de uma RLC em Barra com CSMA-CD.

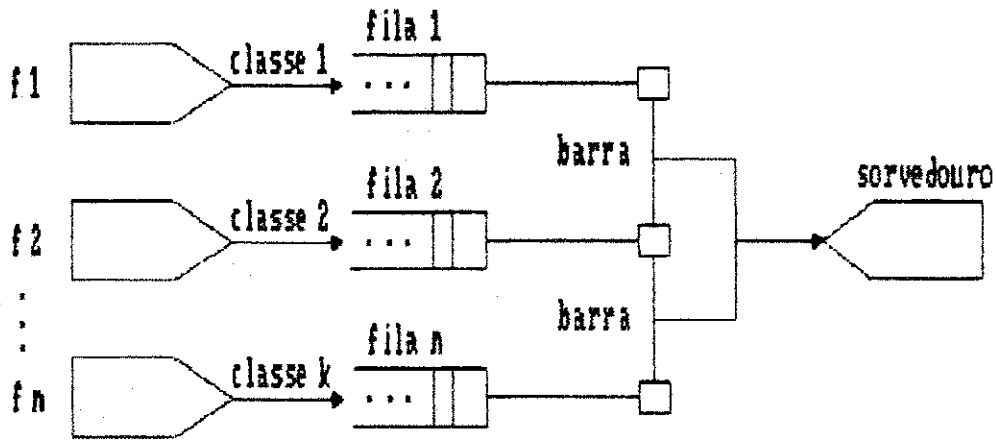


**Fig. 2.4 - Rede Local em Barra com CSMA-CD**

Nessa figura, *fi* (*i*=1,2,...,n) representa a fonte de geração de pacotes em cada interface *i*. Cada interface possui um tipo de classe (k, k<=n) e uma fila de pacotes, cujo comprimento (L) pode ser limitado ou ilimitado.

Uma RLC em Barra com CSMA-CD pode ser modelada por um servidor único, que representa o barramento físico (meio

de transmissão). Nele as interfaces são conectadas e suas informações são transferidas sequencialmente.

No protocolo CSMA-CD, cada interface monitora o meio de transmissão e transmite apenas quando o meio está desocupado [15].

A operação do protocolo exige que antes de transmitir um pacote a interface verifique o estado do meio de transmissão; se este estiver ativo a interface deve esperar. Após o meio se tornar livre, a interface pode iniciar sua transmissão, contudo ela deve continuar monitorando o meio de transmissão durante um periodo equivalente a duas vezes o tempo de propagação do sinal de transmissão (*slot time*) para assegurar-se se há ou não outras interfaces transmitindo.

Uma dada interface pode entrar em colisão durante a parte inicial de sua transmissão, antes que o sinal transmitido por ela tenha se propagado por todas as interfaces no meio e os efeitos daquele sinal se propaguem de volta. Uma vez que a janela de colisão tenha passado, diz-se que a interface adquiriu o meio: colisões subsequentes são evitadas pois as demais interfaces terão sentido o sinal de transmissão e darão preferência à interface que está transmitindo . O tempo para adquirir o meio é baseado no tempo de propagação de ida e volta do sinal de transmissão.

Na ocorrencia de uma colisão, é notificada a interferencia no meio e ligado o sinal de detecção de

colisão e, em seguida, inicia-se o tratamento da colisão. O algoritmo empregado no tratamento de colisão é o algoritmo "Exponencial Binário Truncado" (*truncated binary exponencial backoff algorithm*) [15], que funciona da seguinte forma: primeiro, uma sequência de bits chamada *jam* é transmitida para assegurar que a duração da colisão seja suficiente para ser notada pelas outras interfaces transmissoras envolvidas na colisão. Depois que o *jam* é enviado, a transmissão é encerrada e é preparada uma tentativa de retransmissão num tempo aleatoriamente selecionado. Este tempo é múltiplo do *slot time*. Uma vez que repetidas colisões indicam um meio sobrecarregado, as retransmissões são retardadas para reduzir a carga ao meio (*backing off*). Isto é conseguido expandindo o intervalo no qual o período de espera para retransmissão é selecionado em cada tentativa. A cada tentativa é incrementado o número de tentativas e testado se foi atingido o seu limite. Eventualmente, a transmissão consegue ser efetivada sem colisão, ou as tentativas são abandonados na hipótese de que o meio falhou ou tornou-se excessivamente sobrecarregado (caso em que estoura o limite de tentativas).

O tempo de espera para a n-ésima tentativa de retransmissão é escolhido como um inteiro randômico (r) uniformemente distribuído entre $0 < r < 2^k$, onde $k=min(n,10)$.

De acordo com as considerações acima e a figura 2.4, a interface *i* pode ser modelada conforme a figura 2.5.

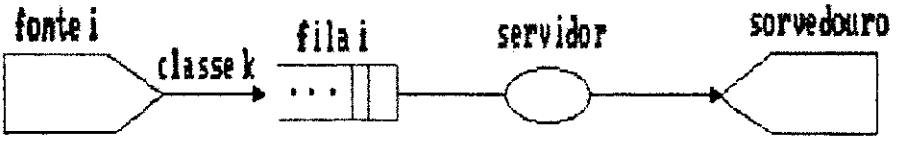


Fig. 2.5 - Modelo da Interface

Neste modelo, em caso de colisão, o pacote que colidiu entra na primeira posição da fila da interface e aguarda um tempo aleatório para que seja feita a tentativa de retransmissão. Portanto, os pacotes presentes na fila de uma interface podem ser pacotes gerados pela fonte ou pacotes que colidiram no meio de transmissão; o que torna a solução analitica, para o modelo da figura 2.5, extremamente difícil de ser realizada.

Neste caso, o tempo de serviço para um pacote que esteja na iminência de ser servido corresponde ac tempo de transmissão do pacote, mais o tempo em que fica esperando até que o meio se torne livre (tempo para adquirir o meio) e, em caso de colisão, acrescenta-se, também, o tempo gasto nas tentativas de retransmissão, além do tempo de retransmissão, conforme o algoritmo de tratamento de colisão

acima mencionado. O tempo para adquirir o meio é uma variável aleatória que depende das características dos tráfegos presentes e do meio de transmissão.

Todo pacote, após ser servido pelo meio de transmissão, entra no nodo sorvedouro, onde é eliminado do modelo, significando que esse pacote foi transmitido pela RLC e entregue ao seu respectivo destino.

### 2.1.4 Modelo do Servidor Único

Um modelo comumente utilizado para a avaliação do desempenho de protocolos de acesso em RLCs é o modelo de Servidor Único mostrado na figura 3.4 [23]. Este modelo pode ser usado para qualquer RLC, independente do tipo de protocolo de acesso ao meio.

Nesse modelo, o servidor incorpora as características do meio de transmissão e do protocolo de acesso da RLC, os fregueses representam pacotes, e *fi* (*i*=1,2,...,n) representa a fonte de geração de pacotes em cada interface *i*. Cada interface possui um tipo de classe (k, k<=n) e uma fila com armazenamento de pacotes, cujo comprimento (L) pode ser limitado ou ilimitado.

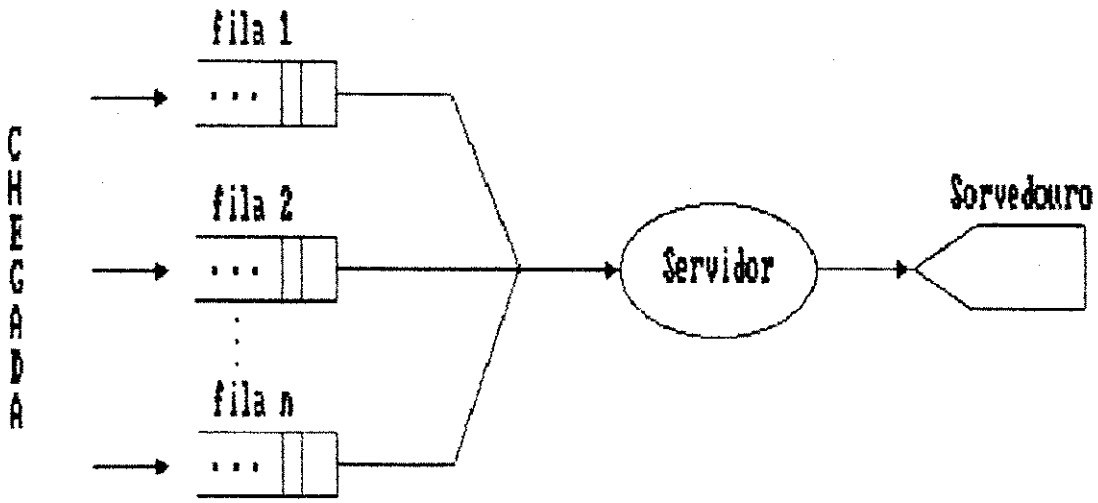


Fig. 2.6 - Modelo do Servidor Único

Um pacote, ao chegar à interface, é transmitido imediatamente se o meio de transmissão estiver disponivel ou, caso contrário, aguarda em fila. Este sistema de fila corresponde, em geral, a uma fila G/G/1. A solução analitica desta fila poderá ser extremamente difícil, dependendo da natureza dos processos de chegada e serviço de fregueses. Na inviabilidade de uma solução analitica, a utilização da técnica de Simulação Digital é geralmente recomendada.

A disciplina de atendimento às filas pelo servidor é estabelecida pelos detalhes operacionais do protocolo de acesso ao meio, conforme descritos anteriormente.

A disciplina de liberação de pacotes em cada inferface pode ser: a) não exaustiva: transmissão de apenas um pacote por vez; b) exaustiva: transmissão de

todos os pacotes existentes na fila da interface; c) limitada: transmissão de um número "x" (x<=L) de pacotes cada vez que a interface detém o meio de transmissão.

A disciplina de escalonamento das filas das interfaces poderá ser de um dos seguintes tipos :

.FCFS (o primeiro que chega é o primeiro que é servido);

.LCFS (o último que chega é o primeiro a ser servido);

.Aleatória.

O modelo do servidor único, aqui definido, prevê a utilização de multiclasses. Uma classe representa o tipo de informação que é transmitida na rede, e pode ser voz, dados, imagens, etc.

Para a distribuição dos tempos de interchegadas de pacotes e do tempo de serviço (tempo de transmissão do pacote) podem ser usadas Funções de Distribuição de Probabilidade (FDPs). Informações sobre essas FDPs são apresentadas no próximo capítulo.

# C A P I T U L O III

## O SIMILE

O SIMILE é um simulador discreto orientado a eventos. Seu projeto e desenvolvimento foram baseados no método de projeto orientado a objetos e na programação orientada a objetos. Sua implementação foi feita na linguagem de programação C++.

O SIMILE foi projetado para avaliar o desempenho de RLCs com Protocolos de Acesso ao Meio, conforme Padrão IEEE 802 (Protocolos com Passagem de Ficha em Anel e em Barra e Barra com Csma-CD), podendo servir como protótipo de simulação de RLCs, por ser facilmente estendido para o estudo e a avaliação de outros protocolos de comunicação. Essa facilidade deve-se às caracteristicas de modularidade, extensibilidade e reutilização de software apresentada pela abordagem orientada a objetos.

O SIMILE oferece a possibilidade de utilização de multiclasses de pacotes (voz, dados, imagens, etc.), que permite a avaliação de desempenho de redes integradas. Oferece, também, a possibilidade de definição do tràfego de entrada do modelo de simulação conforme a aplicação em

questão. Esse tráfego de entrada poderá ser definido das seguintes formas: a) a partir de uma expressão analítica (processo estocástico), e b) através da leitura de um arquivo com dados acerca desse processo estocástico.

No SIMILE são obtidas as medidas de desempenho, consideradas relevantes para cada modelo de simulação, e aquelas pertinentes às aplicações específicas. Ele, também, obtém os intervalos de confiança de um número *n* de execuções.

A seção 3.1 relaciona os elementos relevantes do SIMILE. A seção 3.2 apresenta as medidas de desempenho previstas no SIMILE. A seção 3.3 mostra os parâmetros de entrada para seu processamento. E, finalmente, a seção 3.4 introduz a sua interface.

## 3.1 Elementos do SIMILE

### - Entidades

Entidades são as partes que compõem o sistema. As entidades principais do SIMILE são:

.Meio de Transmissão: transmite as informações de uma interface a outra.

.Protocolo de Acesso ao Meio: controla o acesso ao meio de transmissão, decidindo qual interface, em um dado instante, tem a permissão para transmitir.

.Interface: envia e recebe pacotes.

.Lista de Eventos: armazena os eventos que ocorrem durante a simulação.

.Fila de Pacotes: armazena os pacotes gerados em uma interface.

.Relógio: representa o tempo simulado.

- **Eventos**

Um evento é uma perturbação instantânea que muda o estado do sistema (e por conseguinte, do modelo) [23]. O SIMILE prevê os seguintes eventos:

.Chegada de Pacote;

.Retransmissão de Pacote;

.Fim de Transmissão;

.Fim de Simulação.

O evento "Retransmissão de Pacote" será escalonado apenas quando o protocolo simulado é do tipo CSMA-CD.

- **Estados**

O estado do sistema é a descrição de todas as entidades, atributos e atividades do sistema em um determinado instante de tempo.

Na ocorrência de um evento, o meio de transmissão e/ou uma interface poderá(ão) mudar de estado, conforme segue :

- Meio de Transmissão:

.Ocupado: existe um pacote sendo transmitido;

.Livre: o meio de transmissão está disponivel para a transmissão de pacote.

.Colisão: houve uma colisão de pacotes durante uma transmissão (somente para o protoloco CSMA-CD).

- Interfaces:

.Transmitindo: existe um pacote da interface sendo transmitido no momento;

.Espera: existe pacote(s) na fila da interface a espera que o meio de transmissão fique livre.

Cada interface possui uma fila de pacotes que apresenta os seguintes estados:

.Disponivel: significa que a fila pode receber o pacote gerado na interface;

.Não Disponivel: significa que a fila chegou ao seu limite de armazenamento de pacotes. Os pacotes que chegarem à fila quando ela estiver neste estado serão descartados.

- **Variáveis Aleatórias**

O conjunto de variáveis aleatórias consideradas na implementação do SIMILE são:

.Tempo de Interchegada: intervalo de tempo estimado para a ocorrência do evento "Chegada de Pacote";

.Tempo de Serviço: tempo estimado para a transmissão de um pacote no meio de transmissão.

.Tempo de Retransmissão: tempo para a retransmissão de um pacote que colidiu no meio de transmissão. Esta variável somente é usada quando o protocolo simulado é do tipo CSMA-CD

Os valores atribuídos às variáveis aleátorias, mencionadas acima, podem ser obtidos através das seguintes

FDPs: uniforme, exponencial, normal e geral. Informações sobre essas FDPs se encontram no apêndice A.

## 3.2 Medidas de Desempenho

Com o propósito de tornar o SIMILE versátil na avaliação de desempenho de RLCs, um conjunto abrangente de medidas de desempenho relevantes foi considerado na implementação do SIMILE. Estas medidas são as seguintes:

.Vazão Média: total de pacotes transmitidos no meio de transmissão por unidade de tempo de simulação.

.Tempo Médio de Espera em Fila: tempo médio de espera em fila até a transmissão do pacote pela sub-rede de comunicação.

.Atraso Médio Fim-a-Fim: tempo médio de espera na fila mais o tempo médio de transmissão na sub-rede de comunicação.

.Comprimento Médio de Fila: número médio de pacotes em fila.

.Taxa de Utilização do Meio de Transmissão: fração do tempo que a sub-rede de comunicação utiliza para a transmissão de pacotes.

.Probabilidade de bloqueio nas interfaces: Probabilidade de um pacote que chega a uma interface não ter espaço para ser incluído na sua respectiva fila.

.Cauda da Distribuição do Atraso Fim-a-Fim: estimativa da probabilidade de perda de pacotes de

uma classe. Um pacote de uma classe *i* é considerado perdido se o atraso fim-a-fim excede um valor limite (atributo de classe).

Vale salientar que outras medidas de desempenho podem ser, facilmente, incorporadas ao conjunto considerado acima.

### 3.3 Parâmetros de Entrada do Sistema

Os parâmetros de entrada do SIMILE estão resumidos abaixo. Maiores detalhes sobre estes parâmetros são apresentados na próxima seção. Os parâmetros de entrada são:

- .Número de classes de pacotes;
- .Número de interfaces por classe;
- .Processo de chegada de pacotes para cada classe de pacote, pode ser determinístico, ser obtido através de uma FDP ou de um arquivo contendo os tempos de interchegadas de pacotes;
- .Tempo de serviço para cada classe de pacotes, pode ser determinístico ou ser obtido através de uma FDP;
- .Comprimento da fila de cada interface;
- .Disciplina de escalonamento das filas das interfaces, que pode ser FCFS, LCFS ou Aleatória.
- .Disciplina de liberação de pacotes para cada classe de pacotes, que pode ser: não exaustiva, exaustiva ou limitada. Se for limitada, deverá ser informado o número máximo de pacotes que podem ser transmitidos

pela interface quando ela detém o meio de transmissão;

.Medidas de desempenho de interesse: neste caso, o usuário pode escolher todas as medidas de desempenho do conjunto da seção 3.2 ou um sub-conjunto delas.

.Término da simulação: pode ser pelo tempo total de simulação ou pelo número de pacotes transmitidos pelo meio de transmissão.

.Nível de confiança.

.Sementes para o processo de chegada e para o processo de serviço das *n* execuções do simulador.

.Parâmetros do protocolo de acesso ao meio:

.RLC com Passagem de Ficha:

.Tamanho da ficha livre;

.Nível de prioridade para cada classe de pacotes;

.Limites máximos de tempo de transmissão de pacotes para cada nível de prioridade durante a posse da ficha.

.RLC com CSMA-CD:

.Limite de tentativas de retransmissão;

.*Slot time*: sinal que é enviado para as interfaces da sub-rede de comunicação para avisar que foi dado início a transmissão de um pacote.

.*Jam*: bits adicionais que são transmitidos para garantir que todas as interfaces da sub-rede escutem o sinal de colisão.

## 3.4 Interface do SIMILE

A interface do SIMILE é bastante simples, ela é composta de janelas que mostram opções para serem escolhidas ou, então, solicitam informações necessárias à construção do modelo de simulação.

As figuras 3.1, 3.2, 3.3, 3.4, 3.5 e 3.6 mostram as janelas onde os parâmetros de entrada são informados. Nessas figuras, a direção da seta (-->) representa a próxima janela que será aberta após a entrada da informação solicitada pelo SIMILE ou a escolha da opção correspondente na janela anterior. A escolha de uma opção pode ser feita através de posicionamento do cursor na opção e teclando <ENTER> ou teclando a primeira letra da opção.

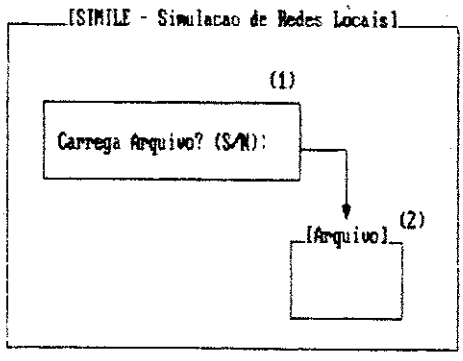


Fig. 3.1 Inicio de Execucao do Simulador

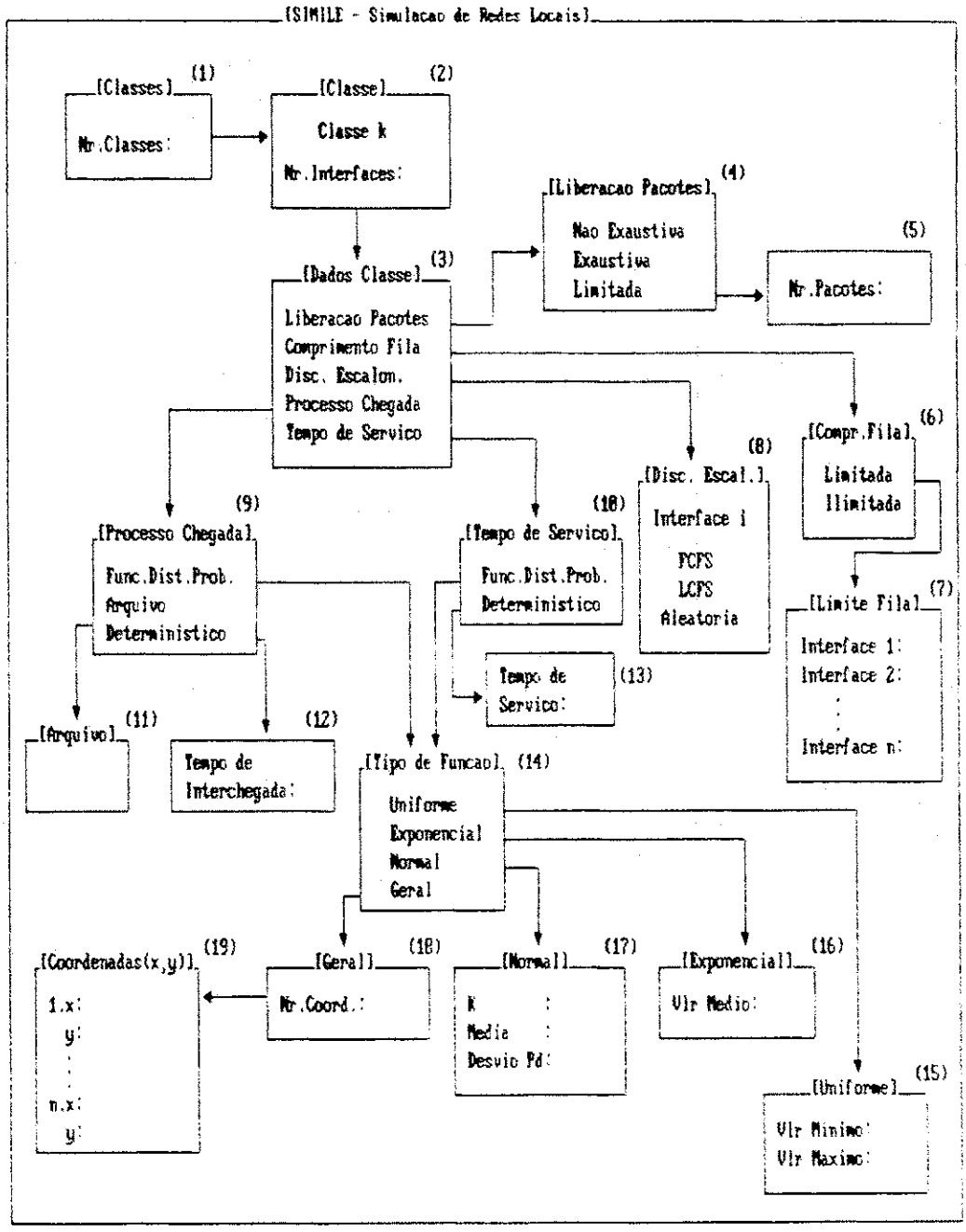


Fig. 3.2 Informacoes sobre as Classes Simuladas

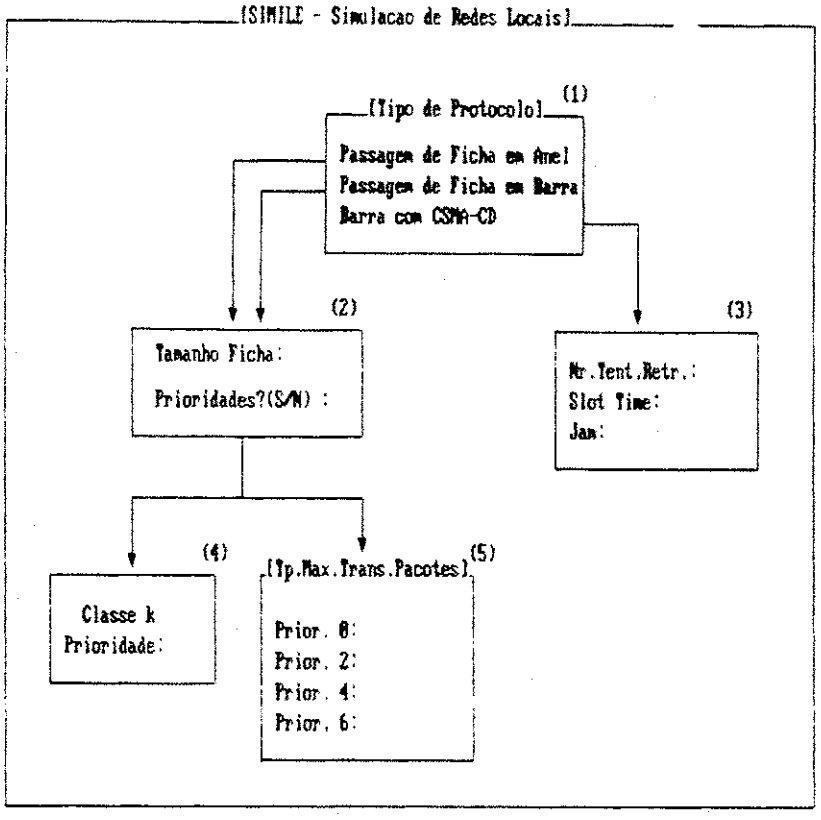


Fig. 3.3 Informacoes sobre os tipos de protocolos

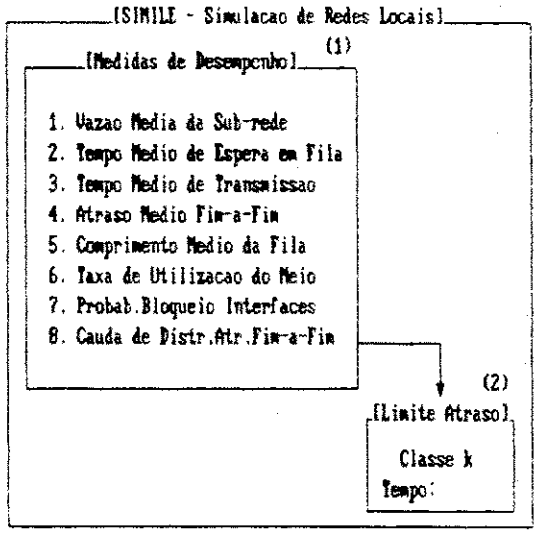


Fig. 3.4 Relacao das Medidas de Desempenho

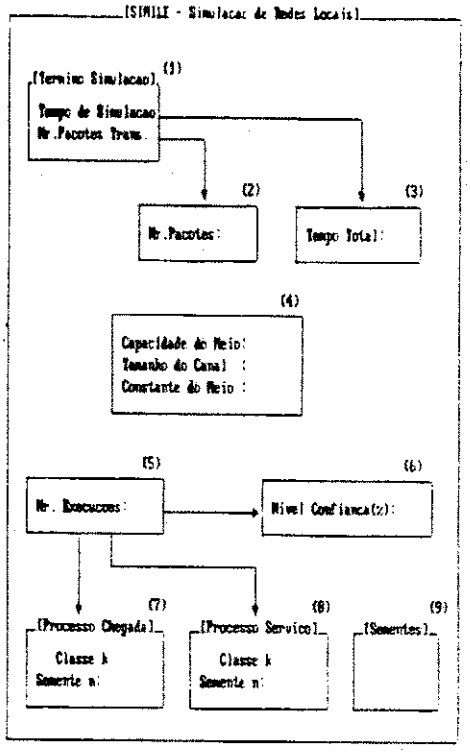


Fig. 3.5 Termino da Simulacao

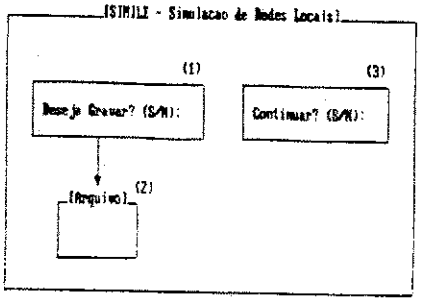


Fig. 3.6 Fim da Entrada de Dados

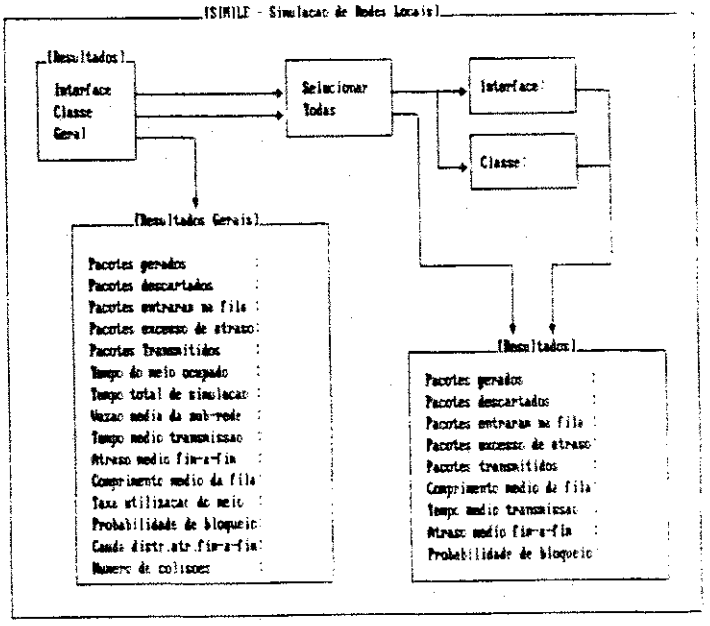


Fig. 3.7 Resultados da Simulacao

A janela (1) da fig. 3.1 é aberta quando o SIMILE é executado, nela o usuário é questionado se deseja ou não carregar um arquivo. Se ele responder que sim, então será aberta a janela (2), onde é solicitado o nome do arquivo que contém as informações necessárias à simulação do modelo do usuário. Caso contrário, será dado início à entrada das informações necessárias à definição do modelo a ser simulado.

Na janela (1) da fig. 3.2 é solicitado o número de classes de pacotes (n) previstas no modelo (voz, dados, imagens, etc.). Em seguida, são solicitadas as informações para cada tipo de classe de pacote. Assim, é aberta a janela (2) (k representa o tipo de classe de pacote, k<=n), onde deve ser informado o número de interfaces pertencentes à classe k. A seguir será aberta a janela (3) contendo opções referentes às informações da classe k. Todas essas opções devem ser escolhidas. Não importando a ordem de escolha feita pelo usuário para essas opções.

Na escolha da primeira opção, "Liberação Pacotes", será aberta a janela (4), que contém as opções "Não Exaustiva", "Exaustiva" e "Limitada". Se a terceira opção for escolhida, então será mostrada a janela (5), onde deve ser informado o número de pacotes que poderão ser transmitidos cada vez que uma interface da classe k detém o meio de transmissão.

Com a escolha da opção "Comprimento Fila", a janela (6) será aberta, com as opções "Limitado" e Ilimitado".

Sendo escolhida a primeira opção, aparecerá a janela (7), onde deve ser informado o tamanho máximo da fila (número máximo de pacotes que poderá permanecer na fila) de cada interface.

Escolhendo a opção "Disc. Escalon.", será aberta a janela (8), onde deve ser selecionado o tipo de disciplina de escalonamento de pacote na fila para cada interface.

Na escolha da opção "Processo Chegada", a janela (9) será aberta, contendo as opções "Func.Dist.Prob.", "Arquivo" e "Deterministico". Se a primeira opção for escolhida, então será solicitado o tipo de FDP na janela (14). De acordo com a FDP escolhida, que pode ser: "Uniforme", "Exponencial", "Normal" ou "Geral", será aberta a janela (15), (16), (17) ou (18), respectivamente. Para a FDP "Geral", a janela (18) solicita o número de coordenadas e a janela (19) solicita as coordenadas x e y. Se for escolhida a segunda opção ("Arquivo"), então será solicitado, na janela (11), o nome do arquivo que contém os tempos de interchegadas de pacotes da classe k. Se a terceira opção for escolhida ("Deterministico"), então será aberta a janela (12) onde deve ser informado o tempo de interchegada de pacotes.

Na escolha da opção "Tempo Serviço", a janela (10) é aberta, ela contém as opções "Func.Dist.Prob." e "Deterministico". Se a primeira opção for escolhida, então será solicitado o tipo de FDP na janela (14). De acordo com a FDP escolhida, que pode ser: "Uniforme", "Exponencial", "Normal" ou "Geral", será aberta a janela (15), (16), (17)

ou (18), respectivamente. Para a FDP "Geral", são solicitados o número de coordenadas através da janela (18) e essas coordenadas na janela (19). Se for escolhida a segunda opção, então será aberta a janela (13) onde deve ser informado o tempo de transmissão de um pacote, que no caso é fixo.

Após a entrada das informações de todas as classes, a janela (1) da fig. 3.3 será aberta, nela estão relacionados os tipos de protocolos de acesso ao meio previstos no SIMILE, que são: Passagem de Ficha em Anel, Passagem de Ficha em Barra e Barra com CSMA-CD.

Os protocolos Passagem de Ficha em Anel e Passagem de Ficha em Barra, conforme foram definidos no capitulo II, funcionam de forma semelhante e suas entradas são idênticas. Essas entradas são: tamanho da ficha, nivel de prioridade para cada classe (janela (4)) e limite máximo de tempo de transmissão de pacotes para cada nivel de prioridade (janela (5)), se existir um mecanismo de prioridade de classes de pacotes, conforme informado na janela (2). Para o protocolo CSMA-CD são socilitados o número de tentativas de retransmissão, o tempo de propagação do sinal de transmissão (*slot time*) e o *jam* na janela (3). Essas janelas são mostradas na fig. 3.3.

A partir do momento em que as classes e o protocolo já estejam definidos, será mostrada uma janela com as opções de Medidas de Desempenho previstas pelo SIMILE, conforme fig. 3.4, para que sejam escolhidas aquelas desejadas pelo

usuário. Se a opção 8 (oito) for escolhida, então será solicitado o tempo máximo de espera de um pacote na fila de pacotes. Quando for encerrada a escolha das medidas de desempenho desejadas, o usuário deve acionar a tecla <ESC> para que continue a entrada das informações restantes.

Em seguida, conforme fig. 3.5, são solicitadas informações sobre o tipo de término da simulação, janela (1). Se o tipo de término for pelo tempo de simulação, então deve ser informado o tempo total de simulação na janela (3). Se for pelo número de pacotes transmitidos pelo meio de transmissão (sub-rede de comunicação), então deve ser informado esse número na janela (2). São solicitadas, também, informações sobre o meio de transmissão na janela (4), essas informações são: capacidade do meio em Mbits/sec, tamanho do canal em quilômetros e constante característica do meio de comunicação.

O usuário pode executar o simulador várias vezes, neste caso, ele deve informar o número de execuções desejadas na janela (5) da fig. 3.5 e o nível de confiança a ser obtido na janela (6). As sementes utilizadas no cálculo dos tempos de interchegada e dos tempos de serviço poderão ser informadas nas janela (7) e (8), ou serem escolhidas de um conjunto apresentado na janela (9) (neste caso o usuário deve entrar com 0 (zero) no lugar da semente), ou ainda, poderão ser escolhidas aleatoriamente pelo SIMILE, caso o usuário não informe essas sementes.

E, para finalizar a entrada dos parâmetros de simulação, é aberta a janela (1) da fig. 3.6 onde o usuário é questionado se deseja gravar as informações do seu modelo. Se ele responder que sim, então será solicitado o nome do arquivo na janela (2).

O usuário tem a opção de continuar ou não com a execução do programa de simulação. Se o usuário confirmar a continuação da execução do programa, então será iniciada a execução da simulação propriamente dita e, quando esta terminar, os resultados obtidos, conforme as medidas de desempenho solicitadas pelo usuário, serão mostrados no vídeo e poderão ser gravadas em um arquivo se o usuário desejar.

# C A P I T U L O IV

## PROJETO ORIENTADO A OBJETOS DO SIMILE

O projeto orientado a objetos, como outras metodologias de projeto orientado a informação, cria uma representação do dominio do problema do mundo real e mapeia este problema em um dominio de solução que é o software [25].

O projeto de um sistema de software orientado a objetos é construído em torno de um conjunto de classes que caracterizam o comportamento dos objetos do mundo real relativos ao sistema. Em geral, uma classe é a descrição de um conjunto de objetos que possuem os mesmos atributos e comportamentos e, um objeto é uma instância especifica de uma classe com comportamento bem definido. Os objetos de cada classe são manipulados através do envio de mensagens. Estas mensagens representam o conjunto de ações que são tomadas sobre o conjunto de objetos.

No Projeto Orientado a Objetos do SIMILE seguiram-se etapas de desenvolvimento de sistemas apresentadas em [25]. Essas etapas são:

1) Identificação das Abstrações de Dados do Sistema;

2) Identificação dos Atributos para cada Abstração de Dados;

3) Identificação das Operações para cada Abstração de Dados;

4) Definição da Comunicação entre Objetos;

5) Teste do Projeto com Cenários, e

6) Aplicação de Herança onde Apropriado.

A seguir são descritas essas etapas:

**1) Identificação das Abstrações de Dados do Sistema**

As abstrações de dados identificadas são as classes de objetos do sistema. No SIMILE foram identificadas, através da descrição de seus modelos, as seguintes classes de objetos:

.Controle

.Protocolo

.Passagem de Ficha

.CSMA-CD

.Meio de Transmissão

.Ficha Livre

.Anel Lógico

.Classe

.Lista de Classes

.Interface

.Lista de Interfaces

.Pacote

.Fila de Pacotes

.Evento

.Lista de Eventos

.Relógio

.Função de Distribuição de Probabilidade

.Função Uniforme

.Função Exponencial

.Função Normal

.Função Geral

As definições das classes e outras informações sobre as mesmas se encontram no protocolo de descrição de classes (apêndice B).

**2) Identificação dos Atributos para Cada Abstração de Dados**

Os atributos tornam-se as variáveis de classe e variáveis de instância para cada classe. Essas variáveis são relacionadas no protocolo de descrição de classes (apêndice B).

**3) Identificação das Operações para Cada Abstração de Dados**

As operações são os métodos de classe e métodos de instância para cada classe. Esses métodos são descritos nos protocolo de descrição de classes (apêndice B).

**4) Definição da Comunicação entre Objetos**

Esta etapa define as mensagens que cada objeto envia. Detalhes desta etapa são encontrados na seção 4.1, que apresenta o modelo cliente-servidor e na seção 4.3 que mostra extensões desse modelo.

**5) Teste do Projeto com Cenários**

Os cenários escolhidos são apresentados na forma de estudos de casos, sendo validados conforme é mostrado no capítulo V.

**6) Aplicação de Herança onde Apropriado**

Nesta etapa, dados e operações comuns são identificados. O objetivo é reutilizar o máximo possivel os dados e/ou métodos já definidos. Desta forma, variáveis e métodos de instância podem ser herdados por novas classes. A herança pode ser visualizada na hierarquia de classes apresentada na seção 4.2.

As etapas definidas acima podem ser repetidas a cada nível de abstração. Assim, refinamentos sucessivos do projeto são feitos. Cada nível de abstração é implementado a um nível mais baixo até um ponto onde a abstração corresponde a um elemento primitivo no projeto.

O restante deste capitulo apresenta seções que detalham as etapas descritas através de representações sugeridas por outros autores [3, 14]. Essas representações são: Modelo Cliente-Servidor, Hierarquia de Classes e Extensões do Modelo Cliente-Servidor. Finalmente, apresenta-se no final deste capitulo, a descrição do funcionamento do SIMILE.

## 4.1 Modelo Cliente-Servidor

O projeto de um sistema orientado a objetos não é realizado apenas em termos dos objetos, mas também dos serviços que eles fornecem a outros objetos. O modelo cliente-servidor 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. [14].

A figura 4.1 mostra a notação usada no modelo cliente-servidor apresentado nas figuras 4.2 e 4.3:

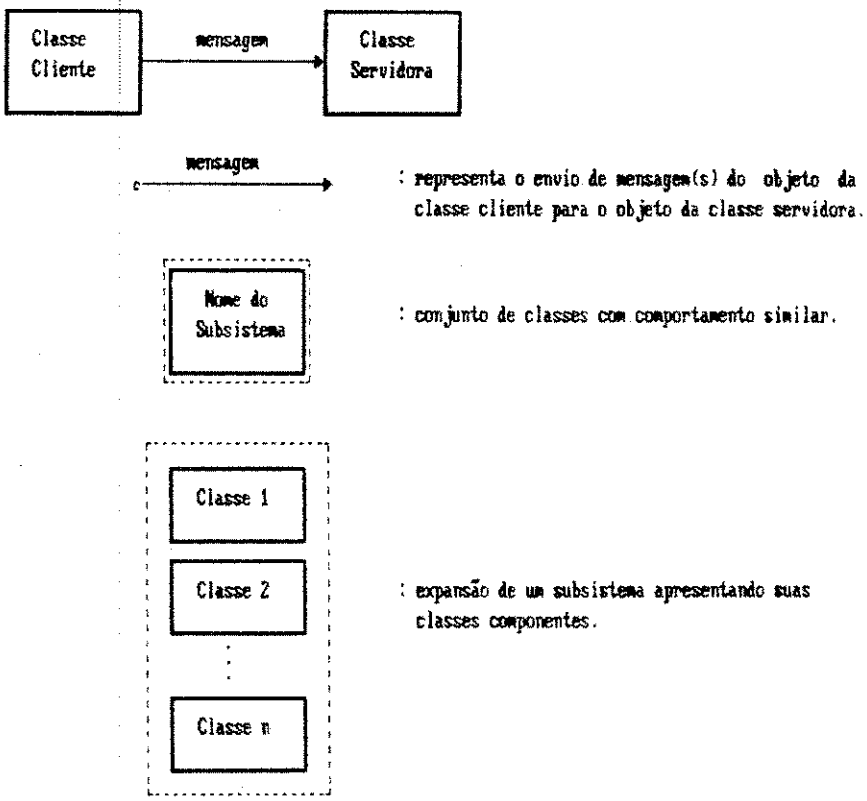


Fig. 4.1 Notação do Modelo Cliente-Servidor

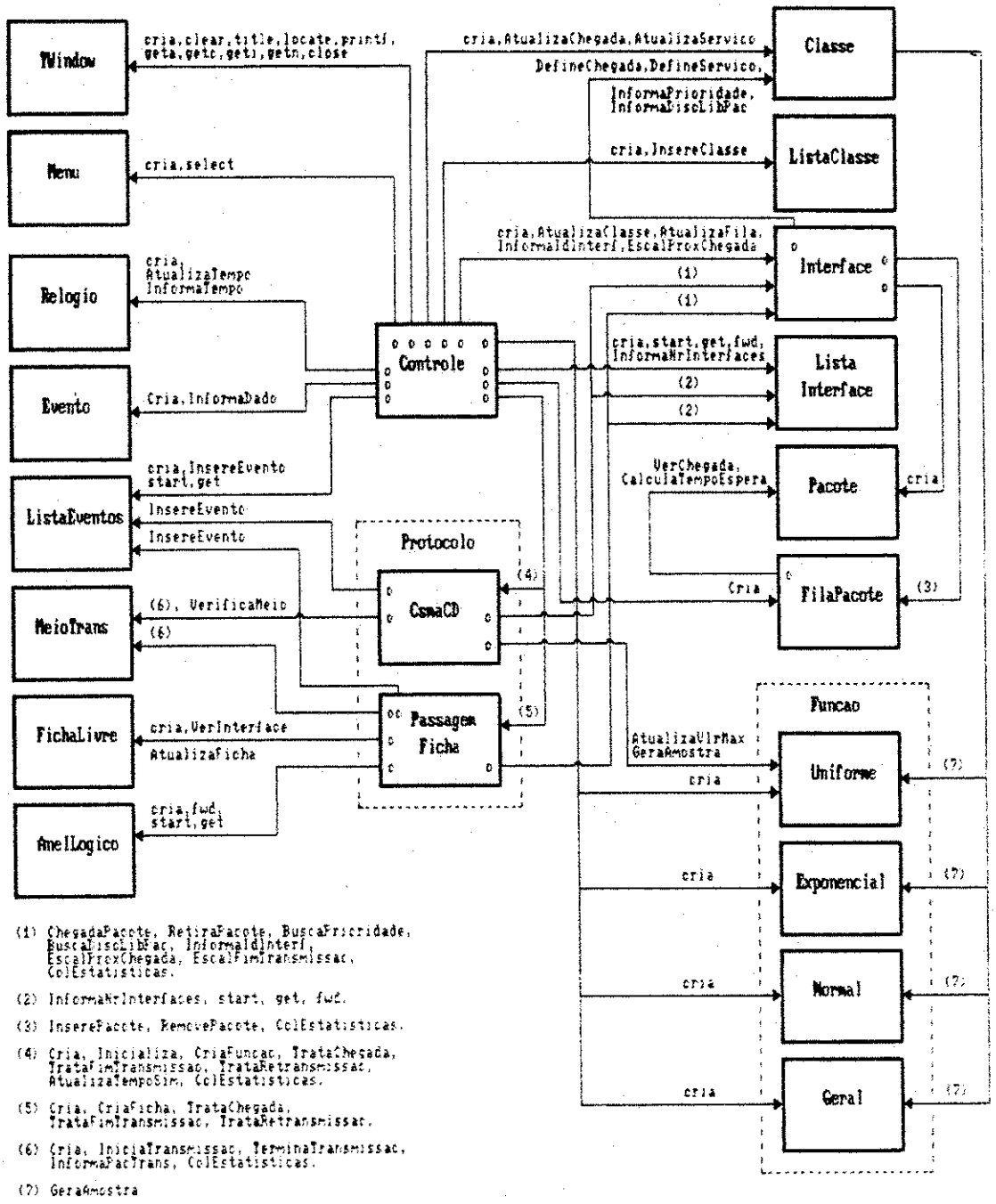


Fig. 4.3 Modelo Cliente-Servidor Expandido

Na figura 4.2 o subsistema "Protocolo" agrega as classes que definem os protocolos de acesso simulados pelo SIMILE, no caso, Passagem de Ficha e CSMA-CD. O subsistema "Função" agrega as classes que definem os tipos de FDPs previstas pelo simulador, que são: Uniforme, Exponencial, Normal e Geral.

A figura 4.3 apresenta o modelo cliente-servidor da figura 4.2 com expansão dos subsistemas "Protocolo" e "Função", mostrando suas classes componentes e relacionamentos específicos de seus objetos com os outros objetos do sistema. O objeto da classe Protocolo gerencia algumas funções que são independentes do tipo de protocolo simulado. Essa facilidade deve-se às características de herança, amarração dinâmica e polimorfismo apresentadas na programação orientada a objetos.

Pode-se observar que os objetos da classe Passagem de Ficha e da classe CSMA-CD são independentes, isto é, o objeto da classe Passagem de Ficha não se comunica com o objeto da classe CSMA-CD e ambos se relacionam com os demais objetos do sistema, o mesmo ocorre com os objetos das classes Uniforme, Exponencial, Normal e Geral.

Com exceção das classes "TWindow", "Menu" "FichaLivre" e "AnelLógico", todas as classes apresentadas nas figuras 4.2 e 4.3 podem constar em projetos de simuladores de RLCs, com pequenas modificações ou mesmo sem nenhuma modificação, por serem independentes do tipo de protocolo de acesso ao meio utilizado. As classes "TWindow"

e "Menu" são necessárias para a interface do sistema. A classe "FichaLivre" é somente utilizada na simulação do Protocolo Passagem de Ficha e a classe "AnelLógico" é, também, somente utilizada na simulação do Protocolo Passagem de Ficha em Barra.

Com a estrutura modular apresentada no projeto orientado a objetos do SIMILE, a implementação de um novo protocolo poderá ser feita através da inclusão de uma nova classe de objetos ao protótipo do SIMILE e da definição dos relacionamentos entre os novos objetos e os já existentes, sem modificar a estrutura básica do projeto. Além do mais, as classes já implementadas poderão ser, também, utilizadas pelo novo protocolo, caracterizando aqui a reutilização de software propiciada pela abordagem orientada a objetos. A reutilização de software é permitida pelos mecanismos de herança e amarração dinâmica, que são oferecidos pela programação orientada a objetos.

Como exemplo podem-se citar as redes ópticas, capazes de suportar aplicações heterogêneas (dados, voz, imagens, etc.) [12]. A simulação dessas redes poderá ser realizada pelo SIMILE apenas com a inclusão de classes de objetos que definem as características especificas de funcionamento dos protocolos de acesso ao meio dessas redes, reutilizando as classes de objetos já implementadas. Basicamente, os seguintes procedimentos deverão ser realizados:

1) Identificação da(s) nova(s) classe(s) de objetos que define(m) as características específicas do novo protocolo;
2) Identificação dos atributos dessas novas classes de objetos;
3) Identificação das operações dessas novas classes de objetos. Essas operações serão, provavelmente, similares às operações dos protocolos previstos nessa primeira versão do SIMILE, da mesma forma que ocorre nos protocolos com Passagem de Ficha e CSMA-CD (vide seção 4.4);
4) Definição da comunicação entre os novos objetos e os já existentes;
5) Implementação dessas novas classes de objetos, não esquecendo que a classe de objetos que representa esse novo protocolo deve ser uma subclasse da classe zProtocolo;
6) Alteração do método CriaProtocolo da classe zControle incluindo a possibilidade de criação de um objeto do novo tipo de protocolo;
7) Teste do SIMILE com dados desse novo protocolo.

No que diz respeito à interface do SIMILE, o novo protocolo deverá ser incluído na janela (1) da fig. 3.3. Também outras janelas poderão ser definidas para a entrada das informações específicas desse protocolo, caso necessário.

Os modelos cliente-servidor apresentados nessa seção mostram os relacionamentos dos objetos do SIMILE de uma forma geral, sem se preocupar com as respostas desses objetos ao receberem as mensagens. Essas respostas estão detalhadas nas extensões do modelo cliente-servidor apresentadas na seção 4.3.

## 4.2 Hierarquia de Classes

A hierarquia de classes pode ser construida usando o projeto top-down ou a decomposição botton-up [31].

No projeto top-down, as estruturas de dados do sistema são modeladas como classes. Essas classes são divididas em subclasses mais especializadas e esse processo é continuado até que uma estrutura hierárquica de classes de objetos seja construida. As classes no topo da hierarquia, também conhecidas como classes base ou superclasses, usualmente incorporam, através das variáveis e dos métodos, caracteristicas que são comuns a todas as classes mais baixas da hierarquia. Essas caracteristicas comuns são herdadas pelas classes mais baixas da hierarquia, também conhecidas como classes derivadas ou subclasses.

Na decomposição botton-up, as subclasses estendem uma classe pai acrescentando maior funcionalidade à classe básica. A classe básica (ou classe pai) incorpora as caracteristicas minimas de suas subclasses e as subclasses herdam essas caracteristicas.

A característica de uma linguagem orientada a objetos que torna possível a implementação da hierarquia de classes é chamada "herança".

A herança é um mecanismo "para expressar a similaridade entre classes, simplificando a definição de classes iguais a outras que já foram definidas. Ela representa generalização e especialização, tornando explícitos os atributos e serviços comuns em uma hierarquia de classe" [9].

A herança permite a reutilização de toda ou parte de uma classe existente na construção de uma hierarquia de componentes reutilizáveis de software [31].

As figuras 4.4, 4.5 e 4.6 mostram a hierarquia de classes do SIMILE. Essas figuras foram baseadas nas representações de hierarquia de classes sugeridas por Pressman em [25]. Nessas figuras, a direção da seta (-->) indica a hierarquia das classes. Como pode ser visto nessas figuras, as subclasses herdam todas ou quase todas as características das superclasses.

As figuras 4.4 e 4.5 apresentam métodos das subclasses que possuem os mesmos nomes de métodos das superclasses (ex.: o método TrataChegada é definido na classe zProtocolo e em suas classes derivadas zPassagemFicha e zCsmaCD), isso indica que os métodos das subclasses redefinem os métodos de suas superclasses, por possuirem funcionamentos diferentes. A característica de amarração dinâmica da programação orientada a objetos determina qual

dos objetos das subclasses receberá a mensagem (método da subclasse ou superclasse), somente em tempo de execução, possibilitando desta forma o polimorfismo.

O polimorfismo permite que uma mesma mensagem seja enviada a todos os objetos dentro de uma classe e suas subclasses, como se cada objeto soubesse como responder a essa mensagem, talvez de uma maneira diferente dos objetos de outras subclasses [31].

Na fig. 4.6, a classe zDList é uma classe pré-definida pela Zortech [32]. A partir dessa classe foram derivadas outras classes que possuem características de uma estrutura do tipo lista.

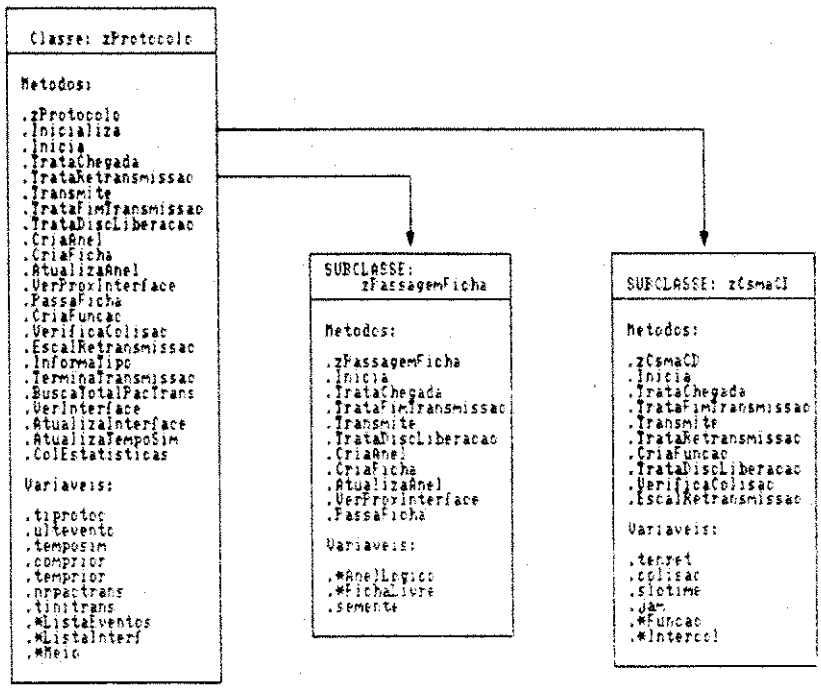


Fig. 4.4 Hierarquia da Classe zProtocolo

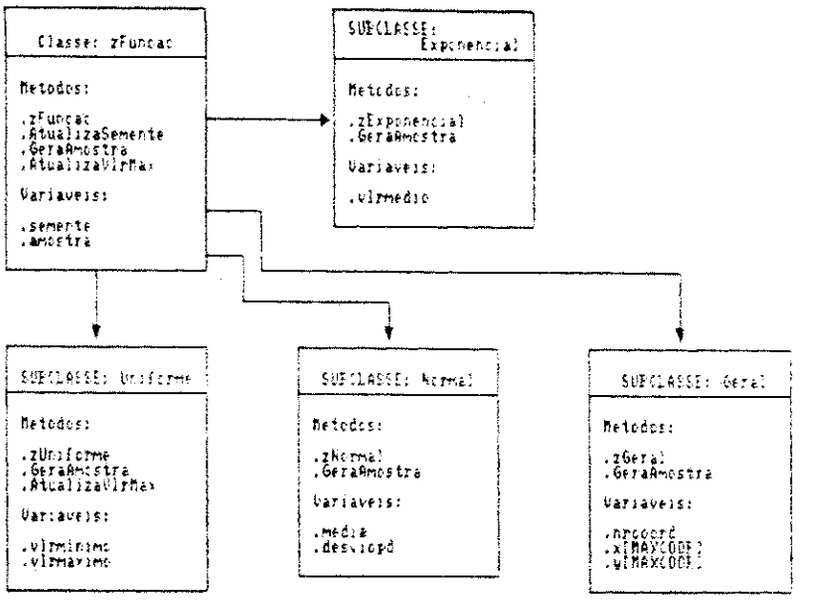


Fig. 4.5 Hierarquia da Classe zFuncao

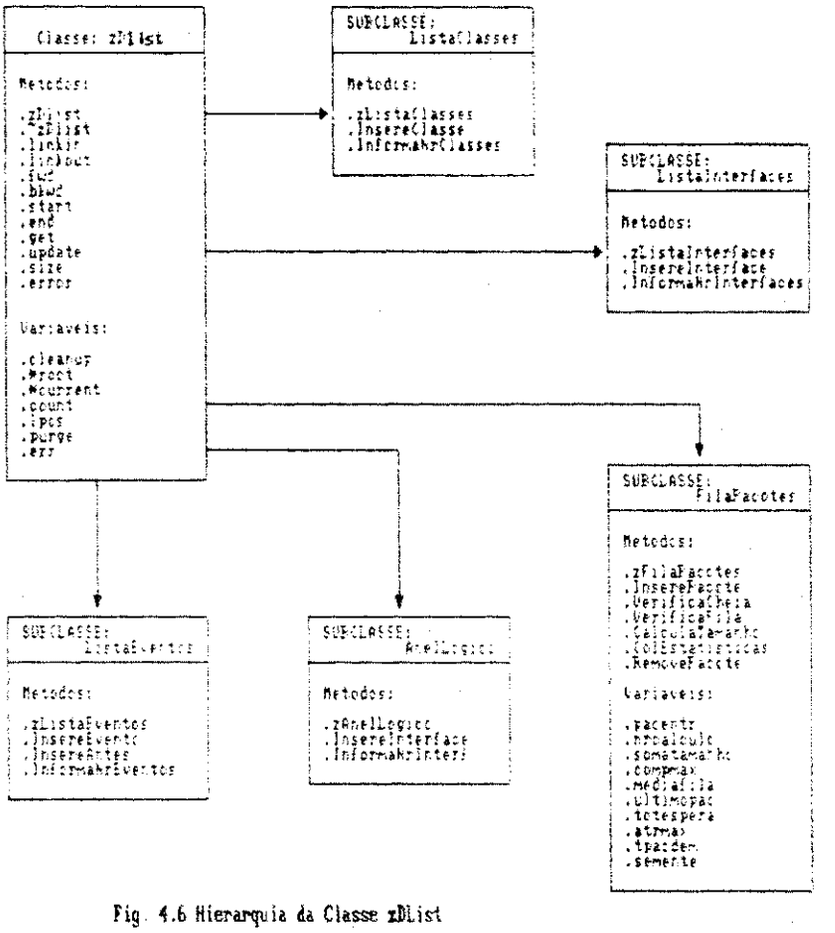


Fig. 4.6 Hierarquia da Classe zDList

As hierarquias de classes apresentadas nas figuras 4.4, 4.5 e 4.6 foram construídas usando a decomposição botton-up, ou seja, classes de objetos com funções gerais, já existentes, são utilizadas na construção de novas classes de objetos com funções específicas.

## 4.3 Extensões do Modelo Cliente-Servidor

As extensões do modelo cliente-servidor mostram as respostas dos objetos ao receberem mensagens. Essas respostas podem ser: envio de mensagens a outros objetos, ativação de métodos da sua própria classe ou atualização de valores associados a seus atributos. Essas extensões se baseiam em diagramas apresentados em [3].

A seguir apresentam-se as extensões do modelo cliente-servidor, relacionados com os principais métodos das classes zSimile, zPassagemFicha e zCsmaCD, por serem estes responsáveis pelo funcionamento do SIMILE.

**- Método Simula**

Na fig. 4.7, a linha pontilhada ao redor dos objetos PassagemFicha e CsmaCD, mostra que o objeto Controle da classe zControle envia mensagens para o objeto PassagemFicha da classe zPassagemFicha ou para o objeto CsmaCD da classe zCsmaCD, dependendo do tipo de protocolo que está sendo simulado. Essas mensagens são métodos das subclasses zPassagemFicha e zCsmaCD que sobrepõem métodos da superclasse zProtocolo (métodos com o mesmo nome). Os objetos PassagemFicha e CsmaCD, apresentados nessa figura,

são criados no método CriaProtocolo da classe zControle com o nome de Protocolo.

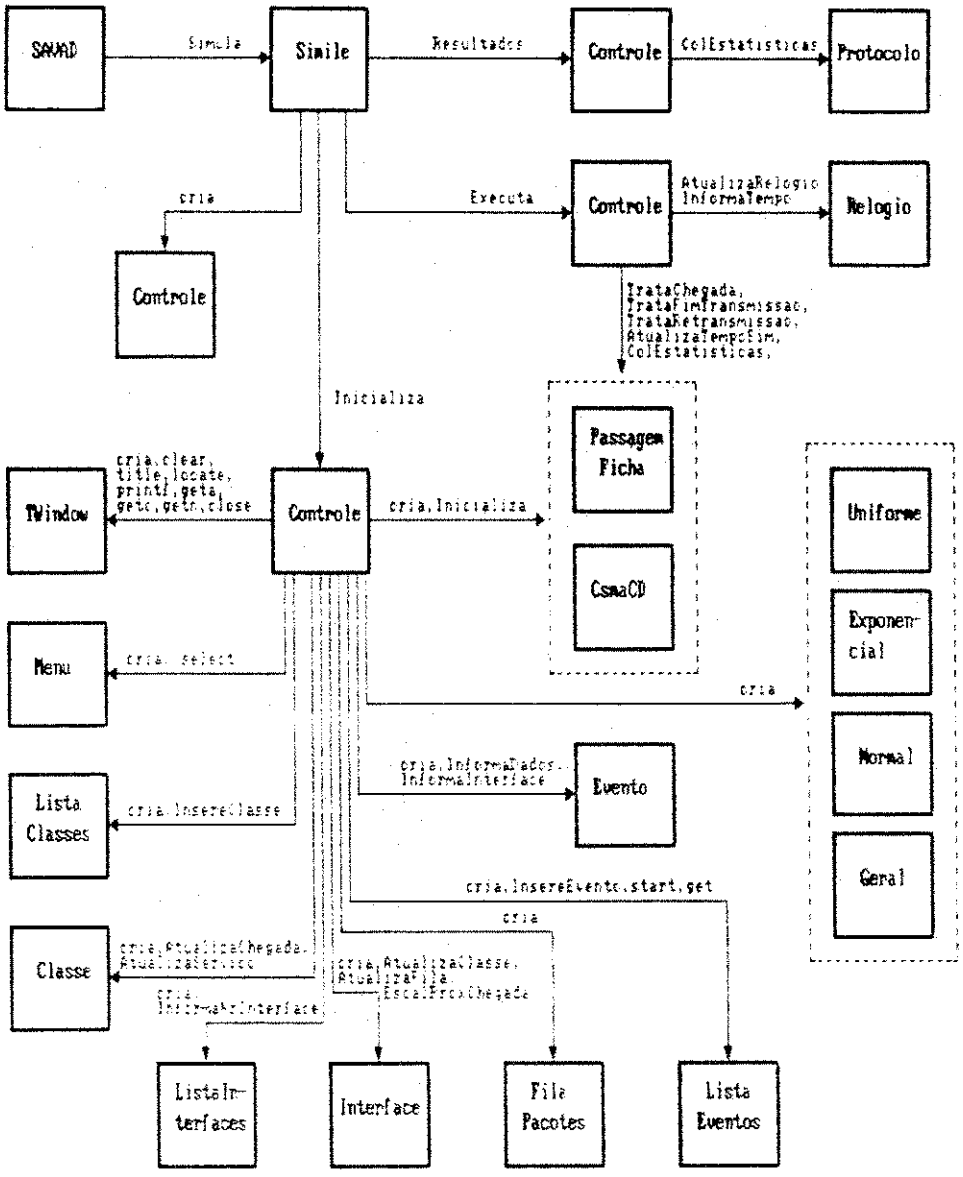


Fig 4.7 Metodo Simula da Classe Simile

A partir do momento em que o objeto Protocolo (que pode ser da classe zPassagemFicha ou da classe zCsmaCD) é criado, as mensagens a ele enviadas são reconhecidas pelo gerenciador da linguagem C++ como sendo da classe zPassagemFicha ou da classe zCsmaCD. Esse mecanismo é conhecido como amarração dinâmica, ou seja, a ligação dos métodos com as classes derivadas é feita em tempo de execução. Ele só é possivel com a utilização do mecanismo de herança entre classes (vide fig. 4.4 da seção 4.2).

O mecanismo de amarração dinâmica ocorre, também, com os métodos das classes zUniforme, zExponencial, zNormal e zGeral. Os objetos Uniforme, Exponencial, Normal e Geral são criados no método CriaFuncao da classe zControle com o nome de Função. (vide fig. 4.5 da seção 4.2).

**- Método TrataChegada**

Na fig. 4.8, a linha dupla diferencia o método TrataChegada da classe zPassagemFicha do método TrataChegada da classe zCsmaCD. As mensagens a objetos, contidas no interior do quadro, estão implementadas apenas no método TrataChegada da classe zPassagemFicha.

A mensagem TrataChegada é enviada pelo objeto da classe zControle para o objeto da classe zPassagemFicha ou para o objeto da classe zCsmaCD quando é escalonado um evento "Chegada de Pacote".

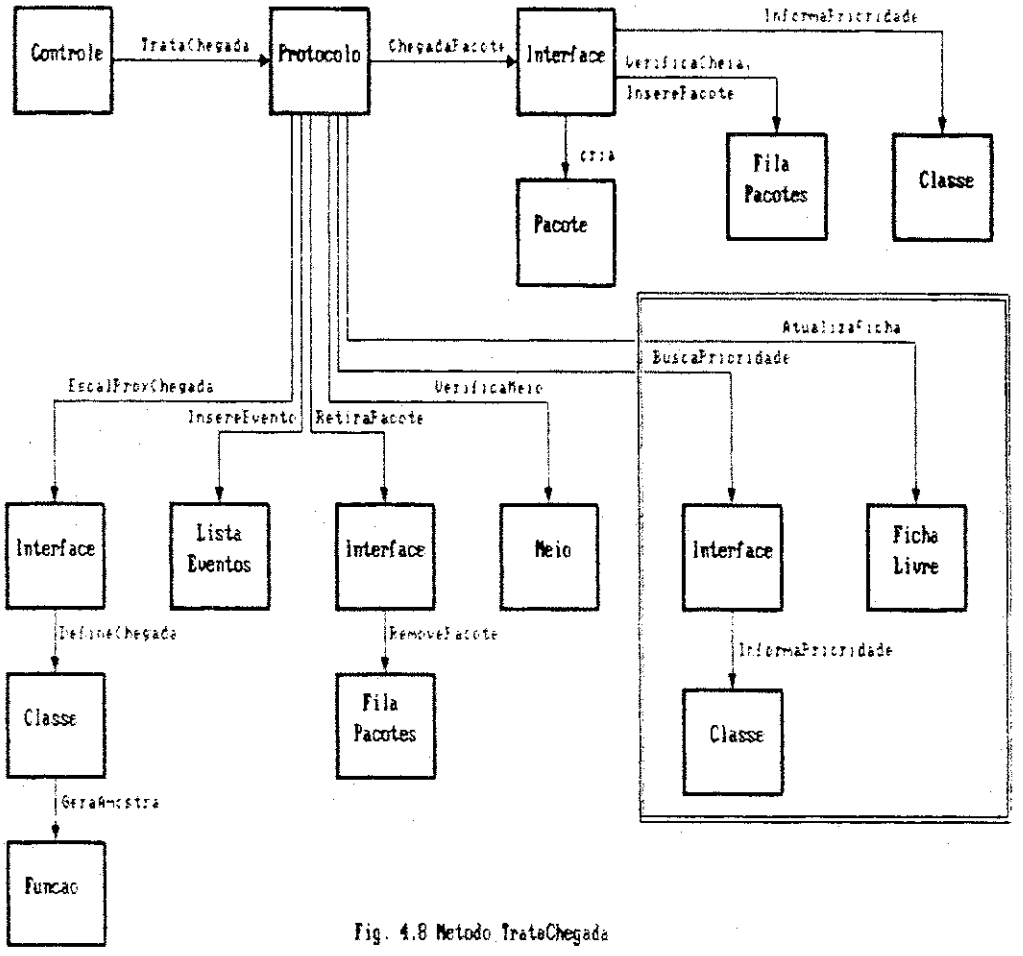


Fig. 4.8 Metodo TrataChegada

- Método TrataFimTransmissão

Na fig. 4.9, a linha dupla diferencia o método TrataFimTransmissão da classe zPassagemFicha do método TrataFimTransmissão da classe zCsmaCD. As mensagens a objetos, contidas no interior do quadro, estão implementadas apenas no método TrataFimTransmissão da classe zPassagemFicha.

A mensagem TrataFimTransmissão é enviada pelo objeto da classe zControle para o objeto da classe zPassagemFicha

ou para o objeto da classe zCsmaCD quando é escalonado um evento "Fim de Transmissão de Pacote".

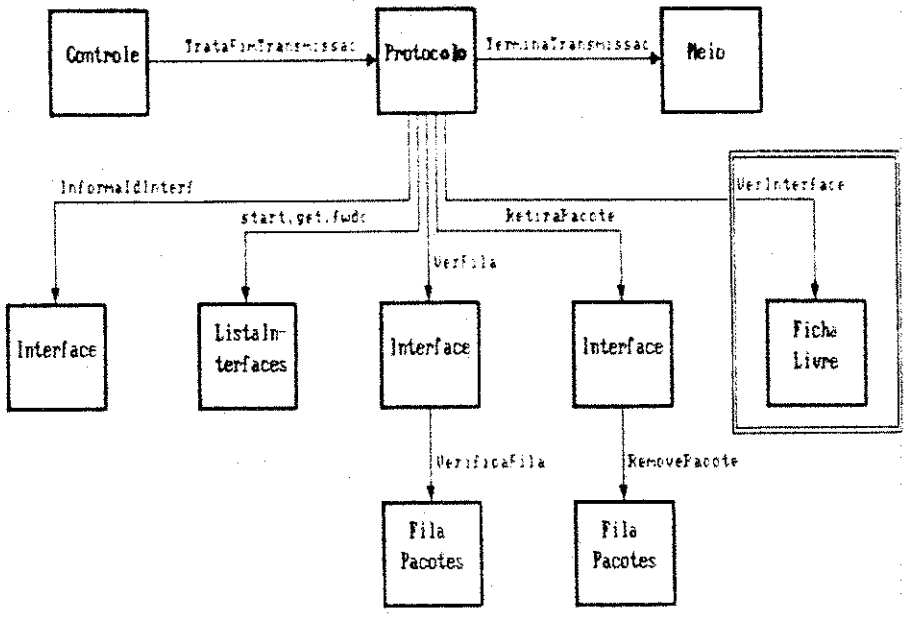


Fig. 4.9 Método TrataFimTransmissao

- Método Transmite

O método Transmite, mostrado na fig. 4.10, é ativado pelos métodos TrataChegada e TrataFimTransmissão das classes zPassagemFicha e CsmaCD, de acordo com o tipo de protocolo que foi criado. Como pode ser visto, por ele ser ativado por métodos da própria classe, não é considerado troca de mensagens.

Apesar do método Transmite da classe zPassagemFicha ser similar ao método Transmite da classe zCsmaCD por enviar as mesmas mensagens ao mesmos objetos do sistema, ele não pode ser implementado uma única vez na classe zProtocolo por possuir controles internos específicos ao tipo do protocolo.

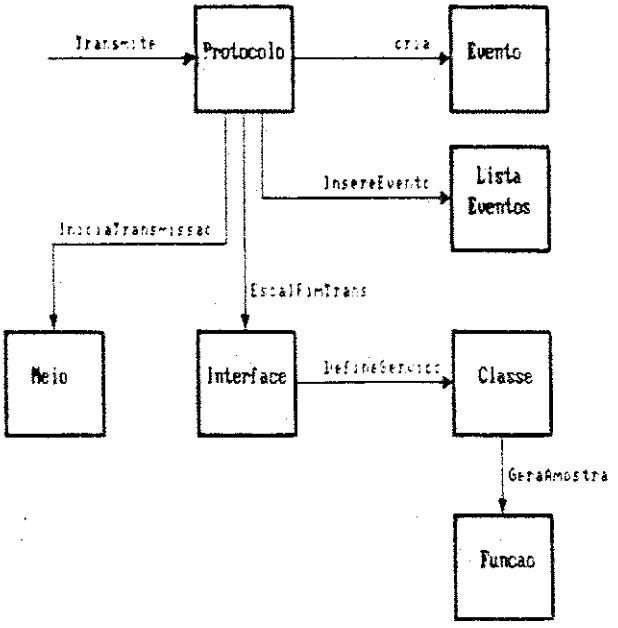


Fig. 4.10 Metodo Transmite

- Método TrataDiscLiberação

Na fig. 4.11, a linha dupla diferencia o método TrataDiscLiberação da classe zPassagemFicha do método TrataDiscLiberação da classe zCsma-CD. Esse método TrataDiscLiberação é ativado pelo método TrataFimTransmissao das classes zPassagemFicha e zCsmaCD, de acordo com o tipo de protocolo que foi criado. Da mesma forma que ocorre com o método Transmite, não é considerado troca de mensagens.

Os métodos TrataDiscLiberação das classes zPassagem Ficha e zCsmaCD, também são semelhantes apenas no envio das mensagens a outros objetos do sistema. No método TrataDiscLiberação da classe zPassagemFicha existe um

controle especial para o mecanismo de prioridade previsto nos protocolos com Passagem de Ficha em Anel e em Barra.

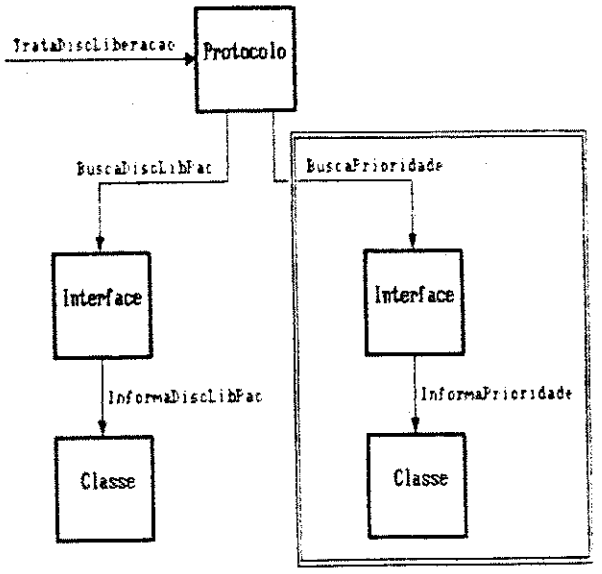


Fig. 4.11 Metodo TrataDiscLiberacao

- **Métodos Específicos ao Protocolo Passagem de Ficha**

O método PassaFicha, mostrado na fig. 4.12, controla a passagem da ficha livre de uma interface a outra. Ele é ativado pelo método TrataFimTransmissão da classe zPassagemFicha.

O método VerProxInterface, mostrado na fig. 4.13, determina a interface que receberá a ficha livre e controla a entrada da interface no anel lógico quando o protocolo sendo simulado é do tipo Passagem de Ficha em Barra. Ele é ativado pelo método PassaFicha da classe zPassagemFicha.

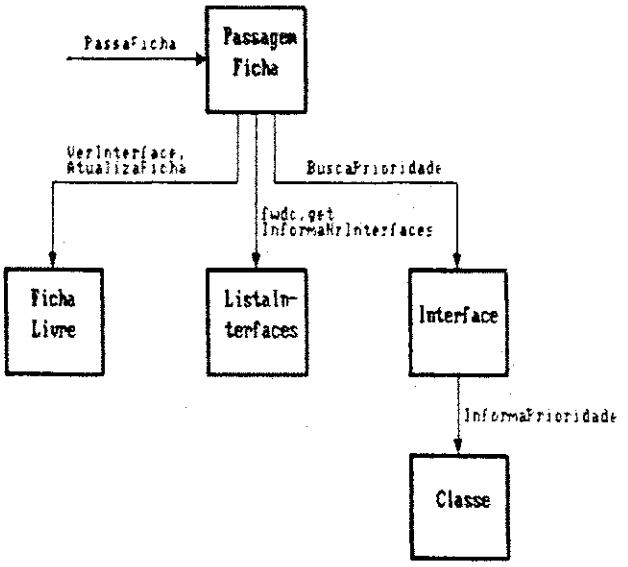


Fig. 4.12 Metodo PassaFicha da Classe zPassagemFicha

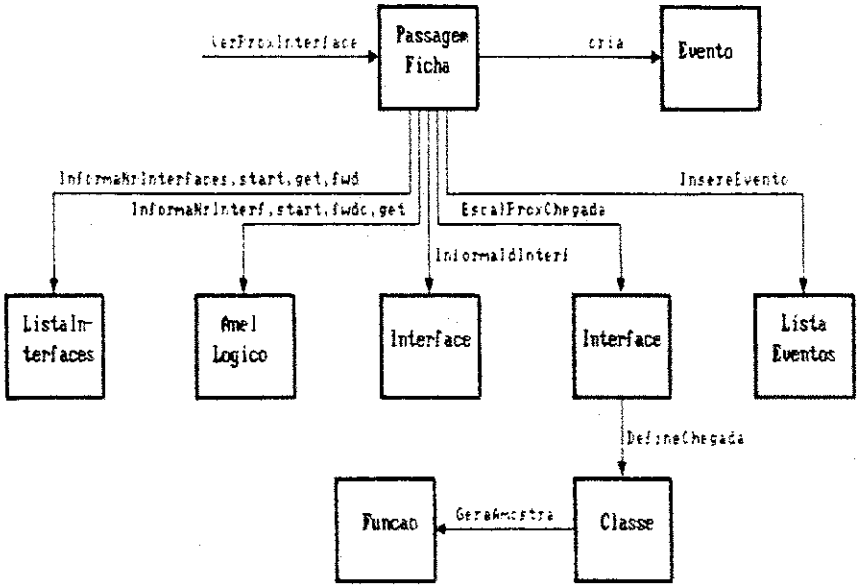


Fig. 4.13 Metodo VerProxInterface da Classe zPassagemFicha

- **Métodos Específicos ao Protocolo CSMA-CD**

O método TrataRetransmissão, mostrado na fig. 4.14, faz o tratamento de retransmissão de pacotes que colidiram no meio de transmissão. O objeto da classe zControle envia a mensagem TrataRetransmissão para o objeto da classe zCsmaCD quando é escalonado o evento "Retransmissão de Pacote".

O método VerificaColisão, mostrado na fig. 4.15, é ativado pelo método Transmite da classe zCsmaCD.

O método EscalRetransmissão, mostrado na fig. 4.16, é ativado pelos métodos TrataChegada e TrataRetransmissão da classe zCsmaCD.

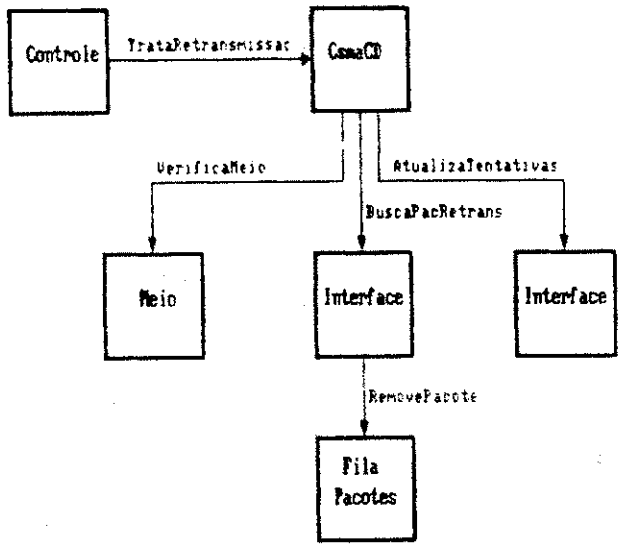


Fig. 4.14 Metodo TrataRetransmissao da Classe zCsmaCD

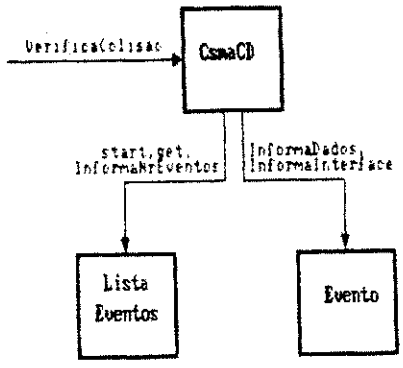


Fig. 4.15 Metodo VerificaColisao da Classe zCsmaCD

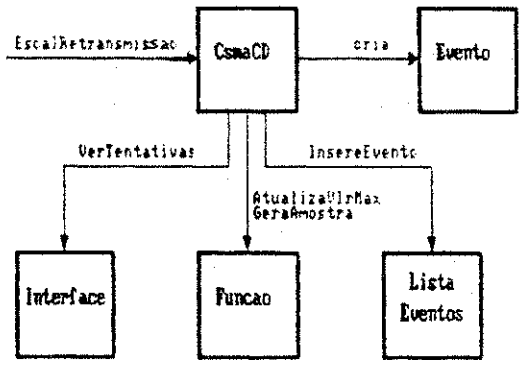


Fig. 4.16 Metodo EscalRetransmissao da Classe zCsmaCD

## 4.4 Descrição do Funcionamento do SIMILE

O SIMILE é um simulador orientado a eventos. Seu funcionamento compõe-se de três etapas: inicialização, processamento e resultados, que são controladas por métodos da classe zControle, ativados pelo programa principal ou pelo método Simula da classe zSimile. Segue uma descrição sucinta dessas etapas. Maiores detalhes sobre os métodos mencionados abaixo, podem ser encontrados no protocolo de descrição de classes (apêndice B).

**1) Inicialização**

Nesta etapa é fornecida a entrada dos dados do modelo a ser simulado, os objetos são criados e suas variáveis são inicializadas com os parâmetros de entrada.

Ela é controlada pelo método Inicializa da classe zControle, que segue os seguintes passos:

.Zera as variáveis da classe zControle;

.Ativa o método CarregaDados se o usuário solicitar que seja carregado um arquivo com as informações do modelo, gravadas anteriormente.

.Ativa os métodos EntraClasse, EntraProtocolo, SelecionaMedidas, EntraTermino e GravaDados, se o usuário deseja entrar com as informações do modelo a ser simulado e não cancelar a execução durante o processamento de um desses métodos.

.O método EntraClasse ativa o método EntraFunção quando o processo de chegada ou o tempo de serviço for calculado através de uma FDP.

.O método EntraProtocolo ativa o método EntraPrioridade quando o tipo de protocolo a ser simulado é Passagem de Ficha, pois para esse tipo de protocolo pode haver um mecanismo de prioridade.

.Ativa o método EntraExecuções para que sejam informados o número n de execuções do simulador, o nivel de confiança desejado e as sementes para as n execuções.

.No final da entrada das informações, se o usuário desejar continuar com a execução do programa simulador, são ativados os métodos CriaListas, CriaProtocolo e CriaClasses para a criação dos objetos necessários ao processamento do simulador.

2) Processamento

Esta etapa faz o processamento completo da simulação. Seus passos podem ser resumidos como segue:

.Escalonamento do evento com menor tempo de ocorrëncia;

.Atualização do relógio simulado;

.Execução das ações relacionadas ao evento escalonado.

Nesta etapa, o controle é feito pelo método Executa da classe zControle, que efetua os seguintes procedimentos:

.Envia a mensagem Inicia para os objetos das classes zProtocolo e zInterface para que sejam inicializadas as variáveis utilizadas nos cálculos estatisticos durante as execuções do simulador.

.Elimina todos os eventos que existirem na lista de eventos (eventos que foram escalonados na execução anterior).

.Ativa o método CriaEventos para que sejam criados eventos "Chegada de Pacote" para dar início a simulação.

.Escalona o primeiro evento da lista de eventos através do envio das mensagens start e get para o objeto ListaEventos.

.Solicita ao objeto Evento, retirado da lista de eventos, informações a seu respeito, são elas: tipo de evento, tempo de sua ocorrência e a interface a ele relacionada.

.Processa os eventos até encontrar a condição de fim de simulação, que pode ser pelo tempo de simulação (evento a ser processado é tipo 4, "Fim de Simulação") ou pelo número de pacotes transmitidos pelo meio de transmissão. Cada evento a ser processado é retirado da lista de eventos através do envio da mensagem linkout para o objeto ListaEventos. O relógio é atualizado com o tempo da ocorrência do evento através da mensagem AtualizaTempo enviada para o objeto Relógio. É, então, enviada para o objeto Protocolo, que pode ser da classe zPassagemFicha ou zCsmaCD, a mensagem TrataChegada se o evento for do tipo 1 ("Chegada de Pacote"), a mensagem TrataFimTransmissão se o

evento for do tipo 2 ("Fim de Transmissão") ou a mensagem TrataRetransmissão se o evento for do tipo 3 ("Retransmissão de Pacote").

.No método TrataChegada é enviada a mensagem ChegadaPacote para a interface que acabou de receber o pacote, esta interface cria um objeto da classe zPacote e o insere na fila de pacotes através da mensagem InserePacote, que é enviada ao objeto FistaPacotes da classe zFilaPacotes. Em seguida, é escalonada a próxima chegada de pacote com o envio da mensagem EscalProxChegada para a interface e um evento "Chegada de Pacote" é criado e inserido na lista de eventos através do envio da mensagem InsereEvento para o objeto ListaEventos. É verificado se o meio de transmissão está disponivel através da mensagem VerificaMeio, se o meio de transmissão estiver disponivel, então o pacote que chegou é transmitido com a ativação do método Transmite.

.No método TrataFimTransmissão é enviada a mensagem TerminaTransmissão para o objeto Meio da classe zMeioTrans, onde é feito o controle de término de transmissão de um pacote. Em seguida, é verificado se a interface ainda tem direito de continuar a transmissão de outros pacotes através da ativação do método TrataDiscLiberação. Se ela tiver esse direito, então outro pacote da mesma interface é

transmitido ativando o método Transmite. Caso contrário, o direito de transmitir é passado para outra interface e um pacote dessa outra interface é transmitido ativando o método Transmite. Nesse método é dado início a transmissão do pacote com o envio da mensagem IniciaTransmissão para o objeto Meio, o evento "Fim de Transmissão" é escalonado através da mensagem EscalFimTrans que é enviada à interface que iniciou a transmissão e, em seguida, esse evento é inserido na lista de eventos através da mensagem InsereEvento que é enviada ao objeto ListaEventos.

.Para o protocolo do tipo CSMA-CD são previstas possíveis colisões de pacotes no meio de transmissão através do método VerificaColisão. Uma colisão ocorre quando o tempo de chegada de um pacote em uma interface, diferente daquela que iniciou a transmissão de pacote, for menor que o tempo atual mais duas vezes o tempo de propagação do sinal de transmissão pela interface que iniciou a transmissão de um pacote, mais o *jam* (tempo atual + 2 * *slot time* + *jam*).

.O evento "Retransmissão de Pacote" só pode ocorrer se o protocolo simulado for do tipo CSMA-CD. No método TrataRetransmissão é feito o controle do número de tentativas de retransmissão. Se o número de tentativas de retransmissão realizados não

excedeu o limite permitido, então o pacote é retransmitido se o meio de transmissão estiver disponivel, se o meio de transmissão não estiver disponível, é escalonada uma nova tentativa de retransmissão através da ativação do método EscalRetransmissão. Se o número de tentativas de retransmissão realizados já excedeu o limite permitido, então o pacote é perdido.

.O objeto Relógio informa o tempo atual através da mensagem InformaTempo que a ele enviada. Em seguida, é enviada a mensagem AtualizaTempoSim para o objeto Protocolo.

.Para finalizar essa etapa, é feita a coleta das estatísticas junto ao objeto Protocolo através do envio da mensagem ColEstatisticas. O objeto Protocolo coleta resultados estatisticos junto aos outros objetos.

3) Resultados

Nesta etapa são obtidos os resultados das simulações executadas, com seus intervalos de confiança. Esses resultados apresentam-se convertidos na unidade de medida definida pelo usuário. Esta etapa é controlada pelo método Resultados da classe zControle.

# CAPITULO V

## IMPLEMENTAÇÃO E VALIDAÇÃO DO SIMILE

A implementação do SIMILE foi feita em microcomputador compatível com IBM-PC e em ambiente MS-DOS. A linguagem de programação utilizada foi C++ versão 2.1 da Zortech [32], por ser uma linguagem de programação orientada a objetos que oferece um ambiente propício para o desenvolvimento de simuladores.

A linguagem de programação C++ foi desenvolvida especificamente para solucionar problemas complexos de simulação.

Os recursos de uma linguagem orientada a objetos (encapsulamento, abstração de dados, herança e amarração dinâmica) são completamente suportados por C++, além de reter os altos níveis de compactação e a velocidade da linguagem de programação C [31].

Segundo Doyle em [11], 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.

Em análises quantitativas e comparativas feitas por Doyle em [11] com algumas linguagens de programação orientadas a objetos e outras não orientadas a objetos, constatou-se que C é a linguagem que possui melhor desempenho. A linguagem de programação C++ ficou em segundo lugar. Com o bom desempenho de C++ aliado às vantagens das linguagens orientadas a objetos, pode-se dizer que C++ tem um desempenho quase tão bom quanto C.

Na implementação do SIMILE foram utilizadas 7 (sete) classes de objetos existentes na biblioteca fonte da Zortech e 21 (vinte e uma) novas classes foram implementadas.

O total de linhas de código implementado pela Zortech na definição das 7 classes de objetos, utilizadas pelo SIMILE, foi de aproximadamente 3.270 e o total de linhas de código que foi necessário na definição das 21 (vinte e uma) novas classes implementadas foi de aproximadamente 5.250. Em ambos os casos foram consideradas as linhas de comentários e as linhas em branco.

Após a implementação do SIMILE, foram feitos testes exaustivos para certificar se este não produzia resultados ou situações indesejáveis, e que todo o mecanismo de troca de mensagens entre seus objetos funcionava corretamente.

Para validar o SIMILE, foram escolhidos quatro estudos de casos que abrangem modelos de RLCs com protocolo de acesso ao meio conforme Padrão IEEE 802. Estes estudos de

casos validam o projeto e a implementação do SIMILE, permitindo desta forma, o seu uso em estudos e projetos de avaliação de desempenho de RLCs.

No primeiro estudo de caso é feita uma comparação do SIMILE com um simulador de RLCs com protocolo de acesso ao meio tipo Passagem de Ficha em Anel, apresentado em [23].

No segundo estudo de caso, os resultados do SIMILE são comparados com os resultados analíticos de um sistema de filas M/M/1.

No terceiro estudo de caso, são comparados resultados do SIMILE, utilizando o modelo de RLCs em Anel com Passagem de Ficha apresentado no capítulo II, com resultados analíticos de um sistema de filas M/G/1 apresentado em [5] e resultados de simulação apresentados em [18].

E, finalmente, no quarto estudo de caso, é feita a validação do modelo de RLCs com protocolo de acesso ao meio tipo Barra com CSMA-CD, através da comparação de resultados do SIMILE com resultados mostrados em [22]. Nas seções que seguem são apresentados esses estudos de casos.

Informações sobre tempos de execução do SIMILE são também abordadas nos estudos de casos mencionados.

## 5.1 Primeiro Estudo de Caso: Comparação do SIMILE com um Simulador de RLCs com Protocolo Passagem de Ficha em Anel

Nesse estudo de caso, os resultados da simulação do modelo de RLCs com protocolo de acesso ao meio tipo Passagem de Ficha em Anel foram comparados com os resultados de um simulador de RLCs, implementado na linguagem de programação C, com as mesmas caracteristicas, conforme apresentado em [23].

As caracteristicas do modelo utilizadas nesse estudo de caso foram as seguintes:

.Disciplina de liberação de pacotes não exaustiva;

.Disciplina de escalonamento de pacotes FCFS;

.Comprimento das filas de pacotes ilimitado;

.Valores dos tempos de interchegada de pacotes obtidos através da FDP Exponencial.

.Tempo de serviço deterministico.

.Sem mecanismo de prioridade nas interfaces.

.O tamanho da ficha livre é desprezado.

.Uma unidade de tempo de simulação (1 uts) corresponde a um microsegundo (1 μs).

O modelo de simulação apresentado em [23] diferencia do modelo do SIMILE em questão, por apresentar apenas uma fonte de geração de pacotes. Para contornar esta diferença, foi multiplicado o tempo médio de interchegada de pacotes pelo numero de interfaces da rede, aumentando assim o tempo médio de interchegada em cada interface. Considera-se neste

estudo de caso duas situações: a primeira, o modelo dispõe de uma única interface e, a segunda, o modelo dispõe de 16 (dezesseis) interfaces.

Para a comparação do SIMILE com o simulador apresentado em [23], de uma RLC com apenas uma interface, foram utilizados os seguintes parâmetros de entrada:

.Número de interfaces: 1;

.Tempo médio de interchegada de pacotes: 2500 uts;

.Tempo de transmissão de um pacote pela sub-rede de comunicação: 1000 uts;

.Número de pacotes transmitidos pelo meio de transmissão: 15000;

.Nível de Confiança: 90%;

.Sementes para a FDP Exponencial, utilizadas para cálculo dos tempos de interchegada: 6427, 21713, 701, 17491 e 26723;

.Sementes para a FDP Uniforme, utilizadas para a escolha da fila de pacotes e necessárias apenas para o simulador apresentado em [23]: 17491, 31, 7993, 43 e 13.

A tabela 5.1 mostra as taxas de utilização do meio de transmissão, obtidas em cinco execuções do SIMILE e do simulador apresentado em [23]. Esta tabela mostra, também, a média amostral dessas execuções.

**Tabela 5.1 Taxa de Utilização do Meio de Transmissão**

| Execução (n) | Taxa de Utilização do Meio de Transmissão | |
|---|---|---|
| | Simulador de [23] | SIMILE |
| 1 | 0,3893 | 0,3893 |
| 2 | 0,3889 | 0,3889 |
| 3 | 0,3891 | 0,3890 |
| 4 | 0,3886 | 0,3886 |
| 5 | 0,3886 | 0,3886 |
| Média Amostral | 0,3889 | 0,3889 |

Nota-se na tabela 5.1, que as diferenças entre as taxas de utilização do meio de transmissão obtidas pelo SIMILE e aquelas obtidas pelo simulador de [23] variam de 0% a 0,02%. Nota-se, também, nessa tabela, que as médias amostrais dos dois simuladores são idênticas.

Os intervalos de confiança obtidos nas execuções do SIMILE variam de 0,07% a 0,32% e, no simulador de [23] variam de 0,09 a 0,32% com nível de confiança de 90%.

A tabela 5.2 mostra os tempos médios de espera de um pacote na fila em unidade de tempo simulado (uts), obtidos em cinco execuções dos simuladores em questão. Esta tabela mostra, também, as médias amostrais das execuções.

**Tabela 5.2 Tempo Médio de Espera em Fila**

| Execução (n) | Tempo Médio de Espera em Fila (uts) | |
|---|---|---|
| | Simulador de [23] | SIMILE |
| 1 | 349,73 | 349,73 |
| 2 | 346,89 | 346,89 |
| 3 | 346,44 | 346,44 |
| 4 | 348,12 | 348,12 |
| 5 | 349,46 | 349,46 |
| Média Amostral | 348,13 | 348,13 |

Na tabela 5.2 oberva-se que os tempos médios de espera em fila obtidos pelo SIMILE são idênticos àqueles obtidos pelo simulador de [23] em todas as execuções. Consequentemente, a média amostral, também, é idêntica.

Os intervalos de confiança obtidos nas execuções do SIMILE e do simulador de [23] são os mesmos e variam de 0,4% a 2,57% com nivel de confiança de 90%.

Para a comparação do SIMILE com o simulador apresentado em [23], de uma RLC com mais de uma interface, foram utilizados os seguintes parâmetros de entrada:

.Número de interfaces: 16;

.Tempo médio de interchegada de pacotes: 40000 uts;

.Tempo de transmissão de um pacote pela sub-rede de comunicação: 1000 uts;

.Número de pacotes transmitidos pelo meio de transmissão: 15000;

.Nível de Confiança: 90%;

.Sementes para a FDP Exponencial, utilizadas para cálculo dos valores dos tempos de interchegada: 6427, 21713, 701, 17491 e 26723;

.Sementes para a FDP Uniforme, utilizadas para a escolha da fila de pacotes e necessárias apenas para o simulador apresentado em [23]: 17491, 31, 7993, 43 e 13.

A tabela 5.3 apresenta as taxas de utilização do meio de transmissão, obtidas em cinco execuções do SIMILE e do simulador apresentado em [23]. Esta tabela mostra, também, a média amostral dessas execuções.

**Tabela 5.3 Taxa de Utilização do Meio de Transmissão**

| Execução (n) | Taxa de Utilização do Meio de Transmissão | |
|---|---|---|
| | Simulador de (23) | SIMILE |
| 1 | 0,3893 | 0,3893 |
| 2 | 0,3889 | 0,3889 |
| 3 | 0,3891 | 0,3890 |
| 4 | 0,3886 | 0,3886 |
| 5 | 0,3886 | 0,3884 |
| Média Amostral | 0,3889 | 0,3888 |

Observa-se na tabela 5.3, que as diferenças entre as taxas de utilização do meio de transmissão obtidas pelo SIMILE e aquelas obtidas pelo simulador de [23] variam de 0% a 0.05%. Nota-se, também, nessa tabela, que as médias amostrais dos dois simuladores são idênticas.

Os intervalos de confiança obtidos nas execuções do SIMILE variam de 0.09% a 0.32% e, no simulador de [23] variam de 0,08% a 0.32% com nível de confiança de 90%.

A tabela 5.4 mostra os tempos médios de espera de um pacote na fila de pacotes em unidade de tempo simulado (uts), obtidos em cinco execuções do simulador de [23] e do SIMILE. Esta tabela mostra, também, as médias amostrais dessas execuções.

**Tabela 5.4 Tempo Médio de Espera em Fila**

| Execução (n) | Tempo Médio de Espera em Fila (uts) | |
|---|---|---|
| | Simulador de (23) | SIMILE |
| 1 | 349,73 | 327,73 |
| 2 | 346,89 | 326,31 |
| 3 | 346,44 | 324,48 |
| 4 | 348,12 | 325,91 |
| 5 | 349,46 | 327,05 |
| Média Amostral | 348,13 | 326,3 |

Nota-se nessa tabela que, as diferenças entre os tempos médios de espera em fila obtidos pelo SIMILE e aqueles obtidos pelo simulador de [23] variam de 0,59% a 6,41%. Essas diferenças foram ocasionados pelo fato do modelo de RLC, simulado pelo SIMILE, apresentar uma fonte de geração de pacotes em cada interface da RLC, enquanto que o modelo de RLC, simulado pelo simulador de [23], apresenta uma única fonte de geração de pacotes para todas as interfaces.

Os intervalos de confiança obtidos nas execuções do SIMILE variam de 0.36% a 1.36.% e, no simulador de [23] variam de 0,4% a 2.57% com nivel de confiança de 90%.

Quando comparado o número de linhas de código do SIMILE com o número de linhas de código do simulador apresentado em [23], verificou-se que o número de linhas de código do SIMILE é de aproximadamente 1.200, enquanto que o simulador apresentado em [23] possui apenas 240 linhas de código. Nessa medição de linhas de código não foram consideradas as linhas de código que definem a interface do SIMILE e também aquelas que definem a interface do simulador de [23], pelo fato do SIMILE possuir uma interface amigável e o outro simulador não possuir.

Um aspecto importante que faz o SIMILE possuir um número de linhas de código maior do que o outro simulador comparado é o fato do SIMILE prever a simulação de modelos com características mais abrangentes do que é previsto no simulador de [23], e essas caracteristicas estarem

implícitas nas linhas de código que implementam as classes que definem o modelo simulado, dificultando a identificação e separação dessas linhas.

Na comparação dos tempos de execução apresentados no SIMILE e no simulador de [23], observou-se os tempos de execução do SIMILE são quase iguais àqueles apresentados em [23], variando no SIMILE de aproximadamente 80" (oitenta segundos) a 135" (cento e trinta e cinco segundos); enquanto que no simulador de [23] o tempo de execução é de aproximadamente 75" (setenta e cinco segundos). A variação dos tempos de execução do SIMILE é devido à passagem da ficha de uma interface para a próxima (os tempos menores são para o caso de se assumir interferência desprezível da transmissão da ficha). Esses tempos foram obtidos em um microcomputador 386 SX com 16 MHz.

## 5.2 Segundo Estudo de Caso: Sistema de Fila M/M/1

Neste segundo estudo de caso é feita uma comparação dos resultados do SIMILE com resultados analíticos de um sistema M/M/1. O modelo utilizado nesta comparação é o modelo de RLC com Passagem de Ficha em Anel. As características consideradas nesse modelo são as seguintes:

.Disciplina de liberação de pacotes não exaustiva;

.Disciplina de escalonamento de pacotes FCFS;

.Comprimento das filas de pacotes ilimitado;

.Tempos de interchegada de pacotes obtidos através da FDP Exponencial.

.Tempos de serviço obtidos através da FDP Exponencial.

.Sem mecanismo de prioridade nas interfaces.

.O tamanho da ficha livre é desprezado.

.Uma unidade de tempo de simulação (1 uts) corresponde a um microsegundo (1 μs).

Os parâmetros de entrada utilizados foram os seguites:

.Número de interfaces: 1;

.Valores dos tempos médios de interchegada de pacotes para cada execução: 1000, 500, 333.33, 250, 200, 166.67, 142.86, 125 e 111 em uts;

.Tempo de transmissão de um pacote pela sub-rede de comunicação: 100 uts;

.Número de pacotes transmitidos pelo meio de transmissão: 15000;

.Nível de Confiança: 95%;

.Sementes para a FDP Exponencial, utilizadas para cálculo dos tempos de interchegada: 74717, 20713, 17727, 31 e 773;

.Sementes para a FDP Exponencial utilizadas para o cálculo dos tempos de serviço: 73053, 20117, 17491, 701 e 151.

A tabela 5.5 mostra os tempos médios de espera de um pacote na fila de pacotes em unidade de tempo simulado (uts), obtidos em execuções do SIMILE, e aqueles obtidos analiticamente pela fórmula a seguir:

$T = \frac{1/\mu .}{1 - \rho}$ onde,

$\mu$ = 1/tempo médio de serviço

$\lambda$ = 1/tempo médio de interchegada de pacotes

$\rho = \lambda/\mu$ .

**Tabela 5.5 Atraso Médio Fim-a-Fim**

| $\rho$ | Atraso Médio Fim-a-Fim (uts) | |
|---|---|---|
| | Solução Analítica | SIMILE |
| 0,1 | 0,11 | 0,11 |
| 0,2 | 0,13 | 0,13 |
| 0,3 | 0,14 | 0,14 |
| 0,4 | 0,17 | 0,17 |
| 0,5 | 0,20 | 0,19 |
| 0,6 | 0,25 | 0,24 |
| 0,7 | 0,33 | 0,32 |
| 0,8 | 0,50 | 0,49 |
| 0,9 | 1,00 | 0,90 |

Na tabela 5.5 observa-se que os atrasos médios fim-a-fim obtidos no SIMILE e aqueles obtidos analiticamente, para um sistema M/M/1, estão bem próximos, variando de 0 a 10%.

Para a obtenção de cada $\rho$ o SIMILE foi executado 5 (cinco) vezes. Nessas execuções foram obtidos intervalos de

confiança que variaram entre 6,79% a 35,42% com 95% de nível de confiança.

## 5.3 Terceiro Estudo de Caso: Sistema de Fila M/G/1

Neste estudo de caso é feita a validação do SIMILE, utilizando o modelo de RLCs em Anel com Passagem de Ficha apresentado no capitulo II, para um sistema de fila M/G/1, através da comparação dos resultados do SIMILE com resultados analiticos mostrados em [5] e resultados de simulação apresentados em [18]. O modelo utilizado nesta comparação é o modelo de RLC com Passagem de Ficha em Anel. As características consideradas nesse modelo são as seguintes:

- .Disciplina de liberação de pacotes não exaustiva;
- .Disciplina de escalonamento de pacotes FCFS;
- .Comprimento das filas de pacotes ilimitado;
- .Valores dos tempos de interchegada de pacotes obtidos através da FDP Exponencial.
- .Tempos de serviço obtidos através da FDP Exponencial (neste tempo é considerado, também, o tamanho da ficha).
- .Sem mecanismo de prioridade nas interfaces.
- .O tamanho da ficha livre é considerado.
- .Uma unidade de tempo de simulação (1 uts) corresponde a um microsegundo (1 μs).

Os parâmetros de entrada utilizados nas execuções do SIMILE foram:

.Número de interfaces da RLC: 10;

.Valores dos tempos médios de interchegada de pacotes, para cada execução: 10000, 5000, 3333.33, 2500, 2000, 1666.67, 1428.6, 1250 e 1111 em uts;

.Tempo médio de transmissão de um pacote pela sub-rede de comunicação: 100 uts;

.Tamanho da ficha livre: 10 uts;

.Número de pacotes transmitidos pelo meio de transmissão: 15000;

.Nível de Confiança: 95%;

.Sementes para a FDP Exponencial, utilizadas para cálculo dos tempos de interchegada: 74717, 20419, 17727, 31 e 773;

.Sementes para a FDP Exponencial, utilizadas para cálculo dos tempos de serviço pela sub-rede: 73053, 20117, 17491, 701 e 151.

A tabela 5.6 mostra os tempos médios de espera de um pacote na fila de pacotes em unidade de tempo simulado (uts), obtidos em execuções do SIMILE, e do simulador apresentado em [18] e, também, obtidos através de cálculos analíticos apresentados em [5].

Observa-se na tabela 5.6 que as diferenças dos tempos médios de espera em fila, obtidos pelo SIMILE e aqueles apresentados em [18] e, também, aqueles apresentados em [5], variam de 0 a 10%.

Tabela 5.6 Tempo Médio de Espera em Fila

| $\rho$ | Tempo Médio de Espera em Fila (uts) | | |
|---|---|---|---|
| | Solução Analítica (5) | Simulação | |
| | | Simulador de (18) | SIMILE |
| 0,1 | 0,65 | 0,65 | 0,60 |
| 0,2 | 0,85 | 0,85 | 0,84 |
| 0,3 | 1,12 | 1,15 | 1,17 |
| 0,4 | 1,40 | 1,55 | 1,69 |
| 0,5 | 1,93 | 2,10 | 2,46 |
| 0,6 | 2,83 | 3,16 | 3,57 |
| 0,7 | 4,64 | 4,90 | 5,20 |
| 0,8 | 8,48 | 8,96 | 8,34 |
| 0,9 | 25,40 | 25,40 | 27,96 |

Para a obtenção de cada $\rho$ o SIMILE foi executado 5 (cinco) vezes. Nessas execuções foram obtidos intervalos de confiança que variaram entre 0,65% a 33,64% com 90% de nivel de confiança.

## 5.4 Quarto Estudo de Caso: Comparação do SIMILE com um simulador de RLCs com CSMA-CD

Neste estudo de caso é feita uma comparação dos resultados do SIMILE com resultados de um simulador apresentado em [22]. O modelo utilizado nesta comparação é o modelo de RLCs CSMA-CD. As caracteristicas consideradas nesse modelo são as seguintes:

.Disciplina de liberação de pacotes não exaustiva;

.Disciplina de escalonamento de pacotes FCFS;

.Comprimento das filas de pacotes ilimitado;

.Tempos de interchegada de pacotes obtidos através da FDP Exponencial.

.Tempos de serviço obtidos através da FDP Exponencial.

Os parâmetros de entrada utilizados foram os seguites:

.Número de interfaces: 16;

.Tamanho do canal: 1 km;

.*Slot*: 10 $\mu$seg;

.Capacidade do meio: 3 Mbits/seg;

.Tamanho máximo do pacote de dados: 4100 bits;

.Tamanho mínimo do pacote de dados: 101 bits;

.Tamanho médio do pacote de dados: 1000 bits;

.Tamanho do cabecalho: 100 bits;

.Tamanho do pacote de reconhecimento: 100 bits;

.Número de pacotes transmitidos: 10.000;

.Atraso na geração do pacote de reconhecimento: 29,65 $\mu$seg;

.Nivel de Confiança: 95%;

.Sementes para a FDP Exponencial, utilizadas para cálculo dos tempos de interchegada: 74717, 20713, 17727;

.Sementes para a FDP Exponencial utilizadas para o cálculo dos tempos de serviço: 73053, 20117, 17491.

Foi considerada, também, uma constante característica do meio de comunicação, que é a capacidade do canal coaxial polipropileno ( $\varepsilon$ = 2,55 em $10^6$Hz). Com essa

constante, obtém-se a velocidade de sinais no meio de comunicação (c), que é a velocidade da luz pela raiz quadrada dessa constante ($v = c/\sqrt{\varepsilon} = 187.867$ km/seg.

Uma unidade de tempo de simulação (uts) equivale ao comprimento do canal (L) pela velocidade de sinais no meio de comunicação (c), então, para o comprimento do canal igual a 1 km tem-se:

uts = L/v = 1/187.867 = 5 $\mu$seg

C = 15 bits/uts

Para o caso de L = 1 km, o tempo de transmissão de um pacote de reconhecimento é igual a 6,667 uts e o tempo de transmissão de um pacote de dados é igual a 74 uts em média.

A tabela 5.7 mostra os atrasos médios fim-a-fim em segundos, obtidos em execuções do SIMILE, e aqueles obtidos pelo simulador apresentado em [22].

**Tabela 5.7 Atraso Médio Fim-a-Fim**

| $\rho$ | Atraso Médio Fim-a-Fim (s) | |
|---|---|---|
| | Simulador de [22] | SIMILE |
| 0,1 | 0,43 | 0,43 |
| 0,2 | 0,50 | 0,48 |
| 0,3 | 0,58 | 0,58 |
| 0,4 | 1,00 | 0,97 |

Na tabela 5.7 observa-se que os atrasos médios fim-a-fim obtidos no SIMILE e aqueles obtidos pelo simulador de [22] estão bem próximos, variando de 0 a 0,3%.

Para a obtenção de cada $\rho$, o SIMILE foi executado 3 (três) vezes. Nessas execuções foram obtidos intervalos de confiança que variaram de 2,04% a 23,7% com nível de confiança de 95%.

# CAPITULO VI

## CONCLUSÕES E SUGESTÕES

O Simulador Reutilizável para Avaliação de Desempenho de Redes Locais, denominado SIMILE, foi projetado para avaliar o desempenho de RLCs com Protocolos de Acesso ao Meio, conforme Padrão IEEE 802. Ele foi desenvolvido baseado na abordagem orientada a objetos e sua implementação foi feita na linguagem de programação C++.

O SIMILE apresenta uma interface bastante simples e uma estrutura modular e reutilizável. Ele fornece medidas de desempenho relevantes para cada modelo simulado e, medidas pertinentes às aplicações especificas consideradas no modelo. Os intervalos de confiança são obtidos automaticamente pelo SIMILE. Ele oferece, também, a possibilidade de definição do tráfego de entrada do modelo conforme a aplicação e a possibilidade da utilização de multiclasses, que permite a avaliação de desempenho de redes integradas (ex.: integração voz e dados).

As caracteristicas de modularidade, extensibilidade e reutilização de software apresentadas na Abordagem

Orientada a Objetos foram exploradas no desenvolvimento do SIMILE para permitir aos projetistas de RLCs, estender com facilidade o seu prototipo com a implementação de outros protocolos de acesso ao meio, diferentes daqueles previstos nesta primeira versão do SIMILE.

A implementação de um novo protocolo de acesso ao meio poderá ser feita através da inclusão de uma nova classe de objetos ao protótipo do SIMILE e da definição dos relacionamentos entre os novos objetos e os já existentes, sem modificar a estrutura básica do projeto. Além do mais, as classes já implementadas poderão ser utilizadas pelo novo protocolo, caracterizando então a reutilização de software, que é permitida pelos mecanismos de herança e amarração dinâmica, oferecidos pela programação orientada a objetos.

O projeto do SIMILE foi dificultado pela falta de representações gráficas adequadas da metodologia de desenvolvimento de sistemas orientados a objetos. Esta dificuldade foi superada com a utilização de sugestões de vários autores [3, 24, 25], em forma de representações gráficas e de textos, que foram adaptadas às necessidades do projeto e desenvolvimento do SIMILE. Dessa forma, o projeto do SIMILE, conforme apresentado no capitulo IV, pode servir, também, como uma referência no desenvolvimento de sistemas orientados a objetos, particularmente no que se refere a simuladores de RLCs.

As comparações feitas entre os resultados de casos de estudo considerados clássicos na literatura de RLCs,

obtidos através do SIMILE, de métodos analíticos e também de outros simuladores conhecidos, validam o projeto e a implementação do SIMILE, garantindo desta forma, o seu uso em estudos e projetos de avaliação de desempenho de RLCs.

Aliado à validação do SIMILE, os resultados dos estudos de casos que envolveram somente simuladores, mostraram que o SIMILE, implementado na linguagem de programação C++, apresentou um tempo de execução de 80" (oitenta segundos), enquanto que o simulador comparado, implementado na linguagem de programação C, apresentou um tempo de execução de 75" (setenta e cinco segundos), quando desprezada a interferência da transmissão da ficha. Portanto, a diferença dos tempos de execução foi de apenas 5" (cinco segundos). Essa constatação está de acordo com os estudos apresentados em literaturas que mostram que o desempenho de C++ é quase tão bom quanto o da linguagem de programação C.

Apresentam-se a seguir algumas sugestões de continuidade do trabalho aqui apresentado:

.Inclusão natural de outros protocolos de acesso ao meio ao protótipo do SIMILE, como por exemplo, redes ópticas que são capazes de suportar aplicações heterogêneas (dados, voz, imagens, etc.) [12]. A simulação dessas redes poderá ser realizada pelo SIMILE apenas com a inclusão de classes de objetos que definem as características específicas de funcionamento dos protocolos de acesso ao meio dessas redes, reutilizando as classes de objetos já implementadas.

.Possibilidade de classes distintas para uma mesma interface.

.Refinamento da apresentação dos resultados do SIMILE de forma gráfica.

.A partir da experiência obtida no desenvolvimento do SIMILE, desenvolver uma metodologia orientada a objetos para a construção de simuladores de sistemas de redes de filas.

.Analisar o SIMILE através de estudos de casos de RLCs com integração de voz e dados.

# APÊNDICE A

## GERAÇÃO DE NÚMEROS E VALORES ALEATÓRIOS

A geração de números aleatórios num computador é geralmente feita através de um algoritmo empregando relações aritméticas e formam uma cadeia de Markov (o próximo número gerado depende apenas do número atual). Geradores aritméticos de números aleatórios são geralmente projetados para produzir números com distribuição uniforme no intervalo [0,1] [23].

Em estudos de simulação outros tipos de distribuição de probabilidade são usualmente necessários. No SIMILE são previstos os seguintes tipos de distribuição de probabilidade: Uniforme, Exponencial, Normal e Geral.

Para a geração de valores aleatórios com Distribuição Uniforme entre dois valores "a" e "b" (a <= b) são seguidos os seguintes passos [13]:

1) Gerar um Número Aleatório r entre 0 e 1 com distribuição uniforme;
2) Usar a transformação: $x = a + r\,(b - a)$.

Para a geração de valores aleatórios conforme a Distribuição Exponencial com valor médio $E[x] = 1/\lambda$ ( é

frequentemente usado para representar a "taxa de chegada"), são seguidos os passos abaixo, conforme Método da Transformação Inversa [13]:

1) Gerar um Número Aleatório r;

2) Usar a transformação: $x = -(1/\lambda)\ln(1 - r)$.

Para a geração de valores aleatórios com Distribuição Normal, são seguidos os passos abaixo [13]:

1) Gerar k Números Aleatórios $(x_1, x_2, \ldots, x_k)$ (o valor conveniente de k é 12);

2) Usar a transformação:

$x = (y - k/2) * S + M$

onde,

$$y = \frac{\sum_{i=1}^{12} x_i - k/2}{(k/12)^{1/2}}$$

S = desvio padrão

M = valor médio

E, para a geração de valores aleatórios com Distribuição Geral, são seguidos os passos abaixo, conforme Método da Aproximação Retangular [13]:

1) Gerar um número aleatório r com distribuição uniforme entre 0 e 1;

2) Achar $x_i$ e $x_{i+1}$, tais que $F(x_i) <= r <= F(x_{i+1})$;

3) Calcular x usando Interpolação Linear:

$$x = x_i + (x_{i+1} - x_i)\left[\frac{r - F(x_i)}{F(x_{i+1}) - F(x_i)}\right]$$

# APENDICE B

## PROTOCOLO DE DESCRIÇAO DE CLASSES

Um protocolo de descrição de classe é a definição completa de todas as propriedades dos objetos da classe (definição, atributos e herança) e a descrição de suas operações [24].

As classes de objetos são identificadas com a letra "z" sendo a primeira letra do seu nome e o objeto de cada classe com o mesmo nome, mas sem a letra "z". Essa convenção é usada pela Zortech [32]. A seguir estão relacionadas as classes de objetos do SIMILE:

**1) zSimile**

Definição:

Esta classe faz a ligação do SIMILE ao SAVAD. Um objeto desta classe é criado pelo SAVAD quando o modelo de simulação, nele especificado, é um modelo de RLCs com protocolos de acesso ao meio, conforme Padrão IEEE 802.

Superclasses: (não possui)

Variáveis: (não possui)

Métodos Públicos:

.zSimile: construtor da classe.

.Simula: cria o objeto Controle da classe zControle e envia as mensagens Inicializa, Executa e Resultados para o mesmo.

.~zSimile: destrutor da classe.

**2) zControle**

Definição:

Esta classe faz o controle de todo o simulador, desde a entrada dos dados até a impressão dos resultados.

Superclasses: (não possui)

Variáveis:

.w0: aponta para a janela principal.

.arqent: nome do arquivo em que os dados de entrada são gravados.

.geral.nrclasses: número de classes.

.geral.tiprotoc: tipo de protocolo simulado.

.geral.nrteret: número máximo de tentativas de retransmissão de um pacote.

.geral.comprior: indica se é para considerar o mecanismo de prioridade de classe, pode ser: 0-sem mecanismo de prioridade, 1- com mecanismo de prioridade.

.geral.slotime: tempo de propagação do sinal de transmissão.

.geral.jam: são bits adicionais que são transmitidos quando ocorre uma colisão de pacotes no meio de

transmissão para garantir que a duração da colisão seja suficiente para ser detectada por todas as interfaces ligadas à sub-rede.

.geral.temprior[4]: tempo máximo de transmissão de pacotes para cada nível de prioridade.

.geral.medida[8]: indica quais as medidas de desempenho solicitadas pelo usuário.

.geral.termino: tipo de término, pode ser: 1-tempo de simulação, 2- número de pacotes transmitidos pelo meio de transmissão.

.geral.nrpactrans: número máximo de pacotes transmitidos durante a simulação.

.geral.tempototal: tempo máximo de simulação.

.geral.nrexec: número de vezes que o simulador será executado.

.geral.nivelconf: nivel de confiança.

.tab_classe[i].nrinterf: número de interfaces da classe i.

.tab_classe[i].prior: nível de prioridade da classe i.

.tab_classe[i].limite[MAXINTER]: comprimento máximo de cada fila das interfaces da classe i.

.tab_classe[i].discescal[MAXINTER]: disciplina de escalonamento de pacote das interfaces da classe i.

.tab_classe[i].dislibpac: disciplina de liberação de pacote da classe i.

.tab_classe[i].nrpacotes: número máximo de pacotes que podem ser transmitidos cada vez que a interface detém o meio de transmissão da classe i.

.tab_classe[i].procheg: tipo de processo de chegada da classe i, pode ser: 1-FDP Uniforme, 2-FDP Exponencial, 3-FDP Normal, 4-FDP Geral, 5-valor deterministico ou 6-arquivo.

.tab_classe[i].tiposerv: tipo de processo de serviço da classe i, pode ser: 1-FDP Uniforme, 2-FDP Exponencial, 3-FDP Normal, 4-FDP Geral ou 5-valor deterministico.

.tab_classe[i].tempinter: tempo fixo de interchegada de pacote da classe i.

.tab_classe[i].tempserv: tempo fixo de serviço da classe i.

.tab_classe[i].atrasomax: tempo máximo de atraso do pacote da classe i.

.tab_classe[i].nomearq: nome do arquivo com tempos de interchegadas da classe i.

.tab_classe[i].semcheg: array com sementes das FDPs do processo de chegada.

.tab_classe[i].semserv: array com sementes das FDPs do processo de serviço.

.tab_classe[i].fdp[j].vlrmin: valor mínimo da FDP Uniforme.

.tab_classe[i].fdp[j].vlrmax: valor máximo da FDP Uniforme.

.tab_classe[i].fdp[j].vlrmedio: valor médio da FDP Exponencial.

.tab_classe[i].fdp[j].media: média da FDP Normal.

.tab_classe[i].fdp[j].desvio: desvio padrão da FDP Normal.

.tab_classe[i].fdp[j].nrcoord: número de coordenadas da FDP geral.

.tab_classe[i].fdp[j].x[MAXCOOR]: coordenadas x da FDP Geral.

.tab_classe[i].fdp[j].y[MAXCOOR]: coordenadas y da FDP Geral.

.ListaClasses: aponta para a lista de classes.

.ListaInterf: aponta para a lista de interfaces.

.ListaEventos: aponta para a lista de eventos.

.Relógio: aponta para o relógio.

Métodos Públicos:

.zControle: contrutor da classe.

.Inicializa: controla a entrada dos dados e inicializa as variàveis da classe.

.Executa: controla as execuções do simulador e coleta as estatísticas de cada execução junto ao objeto Protocolo.

.Resultados: Faz a análise dos resultados das execuções, calcula os intervalos de confiança e mostra os resultados e intervalos de confiança obtidos no vídeo e grava os resultados.

Métodos Privados:

.EntraClasse: controla a entrada dos dados das classes.

.EntraFuncao: controla a entrada das informações necessárias a cada FDP.

.EntraProtocolo: controla a entrada das informações referentes ao protocolo a ser simulado.

.EntraPrioridade: controla a entrada dos níveis de prioridade, caso exista um mecanismo de prioridade de classe.

.SelecionaMedidas: controla a seleção das medidas de desempenho de interesse.

.EntraTermino: solicita o tipo de término de simulação e as informações referentes a cada tipo.

.EntraExecuções: Solicita o número de execuções do simulador, o nível de confiança e as sementes para cada execução.

.TestaDecimal: verifica se o valor decimal digitado está correto.

.MensagemErro: mostra uma mensagem de erro conforme o tipo de erro que ocorreu.

.GravaDados: faz a gravação dos dados de entrada.

.CarregaDados: lê o arquivo com os dados de entrada e armazena nas variáveis da classe zControle.

.InicializaListas: cria as seguintes listas: ListaClasses, ListaInterf e ListaEventos.

.CriaProtocolo: cria o objeto Protocolo, que pode ser da classe zPassagemFicha ou da classe zCsmaCD.

.CriaClasses: cria objetos das classes zClasse e zInterface.

.CriaFuncao: cria o objeto Funcao, que pode ser de uma das seguintes classes: zUniforme, zExponencial, zNormal ou zGeral.

.CriaEventos: cria eventos do tipo 1 ("Chegada de Pacote") para dar início à simulação.

**3) zProtocolo**

Definição:

É uma classe virtual que engloba os aspectos gerais dos protocolos de acesso ao meio, conforme padrão IEEE 802.

Superclasses: (não possui)

Variáveis:

.tiprotoc: tipo de protocolo, pode ser: 1-Anel com Passagem de Ficha, 2-Barra com Passagem de Ficha ou 3-Barra com CSMA-CD.

.ultevento: tipo do último evento ocorrido, pode ser: 1-chegada de pacote, 2-fim de transmissão, 3-retransmissão de pacote ou 4-fim de simulação.

.temposim: tempo total de simulação.

.comprior: indica se existe (1) ou não existe (0) mecanismo de prioridade.

.temprior: tempo máximo de transmissão permitida para cada nível de prioridade, somente em caso de existir mecanismo de prioridade.

.nrpactrans: número máximo de pacotes transmitidos pela interface a partir do momento em que ela adquiriu o meio de transmissão, este controle é feito somente no caso da disciplina de liberação de pacote ser limitada.

.tinitrans: tempo de início de transmissão da interface cada vez que ela detém o meio de transmissão.

.intermeio: aponta para a interface que detém o meio de transmissão.

.ListaEventos: aponta para a lista de eventos.

.ListaInterf: aponta para a lista de interfaces.

.Meio: aponta para o meio de transmissão.

.Pacote: aponta para o pacote que está sendo transmitido.

Métodos Públicos:

.Inicializa: inicializa as variáveis da classe.

.Inicia: método virtual para a inicialização das variáveis utilizadas para cálculos estatísticos a cada execução do simulador.

.TrataChegada: método virtual para o controle do evento "Chegada de Pacote".

.TrataRetransmissão: método virtual para o controle do evento "Retransmissão de Pacote".

.Transmite: método virtual para o controle do processo de transmissão de pacotes.

.TrataFimTransmissão: método virtual para o controle do evento "Fim de Transmissão".

.TrataDiscLiberação: método virtual para o controle do tipo de disciplina de liberação de pacotes.

.CriaFicha: método virtual para a criação da ficha livre no caso do protocolo simulado ser do tipo Anel com Passagem de Ficha ou do tipo Barra com Passagem de Ficha.

.CriaAnel: método virtual para a criação do anel lógico no caso do protocolo simulado ser do tipo Passagem de Ficha em Barra.

.AtualizaAnel: método virtual para a atualização do anel lógico no caso do protocolo simulado ser do tipo Barra com Passagem de Ficha.

.VerProxInterface: método virtual para verificar qual a próxima interface que deve receber a ficha livre.

.PassaFicha: método virtual para o controle da passagem da ficha livre.

.EscalRetransmissão: método virtual para o escalonamento de retransmissão de pacote.

.CriaFunção: método virtual para a criação da FDP Uniforme que gera a amostra para cálculo do tempo de retransmissão do pacote que colidiu.

.VerificaColisão: método virtual para verificar se houve colisão no meio de transmissão.

.InformaColisoes: método virtual para informar o número de colisões ocorridas durante a simulação.

.InformaTipo: informa o tipo de protocolo que está sendo simulado.

.BuscaTotalPacTrans: busca o número total de pacotes transmitidos junto ao meio de transmissão.

.VerInterface: verifica qual a interface que detém o meio de transmissão.

.TerminaTransmissão: termina a transmissão do último pacote antes que a simulação seja encerrada.

.AtualizaInterface: atualiza a interface que detém o meio de transmissão.

.AtualizaTempoSim: atualiza tempo total de simulação.

.ColEstatisticas: coleta estatisticas junto as demais classes do sistema.

**4) zPassagemFicha**

Definição:

Faz o controle da simulação especifica ao funcionamento dos protocolos Anel com Passagem de Ficha e Barra com Passagem de Ficha.

Superclasses:

.zProtocolo

Variáveis:

.AnelLógico: aponta para o objeto da classe zAnelLógico.

.FichaLivre: aponta para o objeto da classe zFichaLivre.

.semente: semente para geração da variável aleatória, que indica qual a próxima interface que deve receber a ficha livre, no caso do protocolo ser do tipo Barra com Passagem de Ficha.

Métodos Públicos:

.zPassagemFicha: construtor da classe.

.Inicia: inicializa as variáveis utilizadas para cálculos estatísticos a cada execução do simulador.

.TrataChegada: controla o evento "Chegada de Pacote".

.Transmite: controla o processo de transmissão de pacotes.

.TrataFimTransmissão: controla o evento "Fim de Transmissão".

.TrataDiscLiberação: faz o tratamento do tipo de disciplina de liberação de pacotes.

.CriaFicha: cria a ficha livre no caso do protocolo simulado ser do tipo Anel com Passagem de Ficha ou do tipo Barra com Passagem de Ficha.

.CriaAnel: cria o anel lógico, no caso do protocolo simulado ser do tipo Barra com Passagem de Ficha.

.AtualizaAnel: atualiza o anel lógico, no caso do protocolo simulado ser do tipo Barra com Passagem de Ficha.

.VerProxInterface: verifica qual a próxima interface que deve receber a ficha livre.

.PassaFicha: faz o controle da passagem da ficha livre para a próxima interface.

**5) zCsmaCD**

Definição:

Faz o controle da simulação especifica ao funcionamento do protocolo em Barra com CSMA-CD.

Superclasses:

.zProtocolo

Variáveis:

.tentret: número de tentativas de retransmissão em caso de colisão.

.colisão: indica se houve colisão, pode ser: 0-não houve colisão, 1-houve colisão.

.slotime: tempo de propagação do sinal de transmissão (*slot time*).

.jam: são bits adicionais que são transmitidos quando ocorre uma colisão de pacotes no meio de transmissão para garantir que a duração da colisão seja suficiente para ser detectada por todas as interfaces ligadas à sub-rede.

.Funcao: aponta para uma FDP Uniforme.

.Intercol: aponta para a interface que colidiu.

Métodos Públicos:

.zCsmaCD: construtor da classe.

.Inicia: inicializa as variáveis utilizadas para cálculos estatísticos a cada execução do simulador.

.TrataChegada: controla o evento "Chegada de Pacote".

.TrataRetransmissão: controla o evento "Retransmissão de Pacote".

.Transmite: faz o controle do processo de transmissão de pacotes.

.TrataFimTransmissão: controla o evento "Fim de Transmissão".

.TrataDiscLiberação: faz o tratamento do tipo de disciplina de liberação de pacotes.

.CriaFuncao: cria uma FDP uniforme que gera amostras para cálculo do tempo de retransmissão do pacote que colidiu.

.EscalRetransmissão: escalona o tempo de retransmissão do pacote que colidiu.

.VerificaColisão: verifica se houve colisão no meio de transmissão.

.InformaColisões: informa o número de colisões ocorridas durante a simulação.

**6) zMeioTrans**

Definição:

Representa o meio de transmissão da rede local.

Superclasses: (não possui)

Variáveis:

.disponivel: indica se o meio de transmissão está disponível (1) ou ocupado (0).

.tinictrans: tempo em que inicia a transmissão de um pacote.

.tmeiocup: tempo total em que o meio de transmissão permanece ocupado.

.tmedio: tempo médio de transmissão de pacotes.

.nrpactrans: número total de pacotes transmitidos durante a simulação.

Métodos Públicos:

.zMeioTrans(): contrutor da classe.

.Inicia: inicializa as variáveis utilizadas para cálculos estatisticos a cada execução.

.VerificaMeio: verifica se o meio de transmissão está disponivel ou não.

.TornaOcupado: torna o meio de transmissão ocupado (disponível=0).

.TornaDisponivel: torna o meio de transmissão disponivel (disponivel=1).

.IniciaTransmissão: inicia a transmissão de um pacote.

.TerminaTransmissão: controla o fim de transmissão de um pacote.

.InformaPacTrans: informa o número de pacotes transmitidos.

.ColEstatisticas: informa as estatísticas obtidas durante a simulação.

**7) zFichaLivre**

Definição:

Representa a ficha livre (ou permissão) que é passada de interface a interface nos protocolos Anel com Passagem de Ficha e Barra com Passagem de Ficha.

Superclasses: (não possui)

Variáveis:

.Interface: aponta para a interface que detém a ficha.

.priorcor: prioridade corrente.

.priores: prioridade reservada.

Métodos Públicos:

.zFichaLivre: construtor da classe.

.Inicia: inicializa suas variáveis.

.AtualizaFicha: atualiza as variáveis da ficha livre.

.VerInterface: verifica a interface que detém a ficha.

.VerPriorCor: verifica a prioridade corrente.

.VerPriorRes: verifica a prioridade reservada.

**8) zAnelLógico**

Definição:

Controla a ordem de passagem da ficha livre entre as interfaces, no caso do protocolo sendo simulado ser do tipo Barra com Passagem de Ficha. Contém apontadores para as interfaces.

Superclasses:

.zDList (classe pré-definida no Zortech)

Variáveis:

.variáveis da classe zDList.

Métodos Públicos:

.zAnelLógico: construtor da classe.

.InsereInterface: insere um apontador para uma interface.

.InformaNrInterf: informa o número de interfaces no anel lógico.

**9) zClasse**

Definição:

Controla as informações gerais sobre as classes.

Superclasses: (não possui)

Variáveis:

.idclasse: identificação da classe.

.procheg: processo de chegada de pacote na interface, pode ser: 1-FDP Uniforme, 2-FDP Exponencial, 3-FDP Normal, 4-FDP Geral, 5-determinístico, 6-arquivo de entrada.

.tiposerv: tipo de cálculo de tempo de serviço, pode ser: 1-FDP Uniforme, 2-FDP Exponencial, 3-FDP Normal, 4-FDP Geral, 5-determinístico.

.prioridade: prioridade da classe.

.disclib: disciplina de liberação de pacotes, pode ser: 1- não exaustiva, 2-exaustiva, 3-limitada por um número de pacotes.

.nrpacotes: número de pacotes que podem ser liberados cada vez que a interface detém o meio de transmissão, somente no caso da disciplina de liberação de pacotes ser limitada (3).

.tempinter: tempo de interchegada de pacote na interface, no caso em que o processo de chegada é definido por um valor detreminístico.

.tempserv: tempo de transmissão de um pacote, no caso em que o tempo de serviço é definido por um valor determinístico.

.Chegada: aponta para a FDP definida para o cálculo do tempo de interchegada de pacotes.

.Serviço: aponta para a FDP definida para o cálculo do tempo de transmissão de um pacote.

Métodos Públicos:

.zClasse: construtor da classe.

.AtualizaSemCheg: atualiza a semente da FDP para o escalonamento da próxima chegada.

.AtualizaSemServ: atualiza a semente da FDP para o cálculo do tempo de serviço.

.AtualizaChegada: atualiza o apontador Chegada.

.AtualizaServiço: atualiza o apontador Serviço.

.DefineChegada: define o intervalo de tempo para a chegada do próximo pacote na interface.

.DefineServiço: define o tempo de serviço para um determinado pacote.

.InformaPrioridade: informa o nível de prioridade da classe, pode ser: 0, 2, 4 ou 6.

.InformaDiscLibPac: informa a disciplina de liberação de pacotes e o número de pacotes que podem ser transmitidos por vez (para disciplina = limitada).

**10) zListaClasses**

Definição:

Armazena apontadores para as classes criadas.

Superclasse:

.zDList (classe pré-definida no Zortech).

Variáveis:

.variáveis da classe zDList.

Métodos Públicos:

.zListaClasses: construtor da classe.

.InsereClasse: insere uma classe na lista.

.InformaNrClasses: informa o número de classes na lista.

**11) zInterface**

Definição:

Representa as interfaces ligadas à rede local.

Superclasse: (não possui).

Variáveis:

.idinterf: identificação da interface.

.discescal: disciplina de escalonamento de pacotes na interface, pode ser: 1-FCFS, 2-LCFS ou 3-Aleatória.

.tentret: número de tentativas de retransmissão de um pacote que colidiu.

.totalpacotes: número total de pacotes gerados pela interface (menos os descartados).

.pacdescar: número total de pacotes descartados pela interface.

.totserv: tempo total de transmissão de pacotes da interface.

.t_medesp: somatória das médias de espera em fila obtidas nas n execuções do SIMILE.

.t_medtrans: somatória das médias dos tempos de transmissões de pacotes obtidas nas n execuções do SIMILE.

.t_compmed: somatória dos comprimentos médios da fila obtidos nas n execuções do SIMILE.

.t_probloq: somatória das probabilidades de bloqueio obtidas nas n execuções do SIMILE.

.t_pactrans: somatória dos totais de pacotes obtidos nas n execuções do SIMILE.

.t_atraso: somatória dos atrasos médio fim-a-fim obtidos nas n execuções do SIMILE.

.Classe: aponta para a classe que a interface pertence.

.FilaPacote: aponta para sua fila de pacotes.

Métodos Públicos:

.zInterface: construtor da classe.

.Inicia: inicializa as variáveis utilizadas para cálculos estatisticos durante cada execução do simulador.

.AtualizaClasse: atualiza o apontador da classe.

.AtualizaFila: atualiza o apontador da fila de pacotes.

.InformaIdInterf: informa a identificação da interface.

.ChegadaPacote: controla o processo de chegada de pacote na interface.

.EscalProxChegada: faz o escalonamento da próxima chegada de pacote na interfce.

.EscalFimTrans: faz o escalonamento do fim de transmissão do pacote.

.VerFila: verifica se a fila de pacotes está vazia.

.BuscaPacote: busca um pacote na fila de pacotes para ser transmitidos.

.RetiraPacote: retira um pacote de sua fila de pacotes.

.BuscaPacRetrans: busca o pacote a ser retransmitido.

.BuscaPrioridade: busca o nivel de prioridade junto à classe que pertence.

.BuscaDiscLibPac: busca disciplina de liberação de pacotes e o número de pacotes que podem ser transmitidos cada vez que a interface detém o meio de transmissão, junto à classe que pertence.

.BuscaCompFila: busca o comprimento da fila de pacotes.

.VerTentativas: informa o número de tentativas de transmissão feita por um pacote na interface.

.AtualizaTentativas: atualiza o número de tentativas de retransmissão de um pacote pela interface.

.ColEstatisticas: coleta as estatisticas calculadas pela interface durante a simulação.

.ColTotais: coleta as estatisticas calculadas pela interface durante as n execuções do SIMILE.

**12) zListaInterfaces**

Definição:

Armazena apontadores para as interfaces.

Superclasse:

.zDList (classe pré-definida no Zortech).

Variáveis:

.variáveis da classe zDList.

Métodos Públicos:

.zListaInterfaces: construtor da classe.

.InsereInterface: insere uma interface na lista.

.InformaNrInterfaces: informa o número de interfaces na lista.

**13) zPacote**

Definição:

Representa o pacote que chega na interface.

Superclasses: (não possui)

Variáveis:

.instcheg: instante de chegada do pacote

.prioridade: nível de prioridade do pacote.

Métodos Públicos:

.zPacote: construtor da classe.

.VerPrioridade: verifica a prioridade do pacote.

.VerChegada: verifica o instante de chegada do pacote.

.CaculaTempoEspera: calcula o tempo de espera do pacote.

**14) zFilaPacotes**

Definição:

Armazena apontadores para os pacotes gerados nas interfaces.

Superclasses:

.zDList (classe pré-definida no Zortech)

Variáveis:

.variáveis da classe zDList.

.pacentr: número total de pacotes que entraram na fila.

.somatamanho: somatória do tamanho da fila.

.nrcalculo: número de vezes que é calculado o tamanho da fila.

.compmax: número máximo de pacotes que podem permanecer na fila.

.totespera: número acumulativo do tempo em que cada pacote fica na fila antes de ser transmitido.

.atrmax: tempo máximo de espera (atraso) de um pacote na fila.

.tpacdem: total de pacotes com tempo de espera > tempo máximo de atraso permitido.

.semente: semente da FDP Uniforme para cálculo da posição do pacote na fila de pacotes quando a disciplina de escalonamento de pacote é aleatória.

Métodos Públicos:

.zFilaPacote: construtor da classe.

.Inicia: inicializa as variáveis utilizadas para cálculos estatísticos durante cada execução do simulador.

.InserePacote: insere um pacote na fila.

.VerificaCheia: verifica se a fila está cheia.

.VerificaFila: verifica se a fila não está vazia, caso não esteja, retorna o instante de chegada do próximo pacote.

.CalculaTamanhoFila: calcula somatamanho e nrcalculo.

.BuscaPacote: informa o pacote que está na iminência de ser servido.

.MudePosiçãoPacote: muda a posição do pacote que colidiu para o início da fila.

.RemovePacote: remove um pacote da fila.

.InformaCompFila: informa o comprimento máximo da fila.

.ColEstatísticas: informa os valores calculados durante a simulação.

**15) zEvento**

Definição:

Representa um evento.

Superclasses: (não possui)

Variáveis:

.tipo: tipo de evento, pode ser: 1-Chegada de Pacote, 2-Fim de Transmissão, 3-Retransmissão de Pacote ou 4-Fim de Simulação.

.tempo: tempo de ocorrência do evento.

.Interface: aponta para a interface relacionada ao evento.

Métodos Públicos:

.zEvento: construtor da classe.

.InformaDados: informa dados sobre o evento.

.InformaInterface: informa a interface que o evento pertence.

**16) zListaEventos**

Definição:

Armazena apontadores para os eventos que ocorrem durante a simulação.

Superclasses:

.zDList (classe pré-definida no Zortech)

Variáveis:

.variáveis definidas em zDList.

Métodos Públicos:

.zListaEventos: contrutor da classe.

.InsereEvento: insere um evento na lista.

.InsereAntes: insere um evento de acordo com o tempo de sua ocorrência.

.InformaNrEventos: informa o número de eventos na lista.

**17) zRelógio**

Definição:

Representa o tempo simulado.

Superclasses: (não possui)

Variável:

.tempo: tempo atual da simulação.

Métodos Públicos:

.zRelógio: construtor da classe.

.Inicia: inicializa o tempo do relógio.

.AtualizaTempo: atualiza o tempo corrente.

.InformaTempo: informa o tempo corrente.

**18) zFunção**

Definição:

É uma classe virtual que representa uma FDP.

Superclasses: (não possui)

Variáveis:

.semente: semente usada no càlculo do número aleatòrio.

.amostra: amostra gerada por uma FDP.

Métodos Públicos:

.zFunção: construtor da classe.

.GeraAmostra: método virtual para a geração da amostra, conforme a FDP informada pelo usuário.

.AtualizaVlrMax: método virtual que atualiza o valor máximo da FDP Uniforme para cálculo do tempo de retransmissão do pacote que colidiu.

.AtualizaSemente: atualiza a semente a cada execução do simulador.

**19) zUniforme**

Definição:

Representa a FDP Uniforme.

Superclasses:

.zFunção

Variáveis:

.vlrminimo: valor minimo.

.vlrmáximo: valor máximo.

Métodos Públicos:

.zUniforme: construtor da classe.

.GeraAmostra: gera uma amostra conforme FDP Uniforme.

.AtualizaVlrMax: atualiza o valor máximo para escalonamento do tempo de retransmissao do pacote, quando o protocolo for do tipo CSMA-CD e houver uma colisão.

**20) zExponencial**

Definição:

Representa a FDP Exponencial.

Superclasses:

.zFunção

Variáveis:

.vlrmédio: valor médio.

Métodos Públicos:

.zExponencial: construtor da classe.

.GeraAmostra: gera uma amostra conforme FDP Exponencial.

**21) zNormal**

Definição:

Representa a FDP Normal.

Superclasses:

.zFunção

Variáveis:

.média: parâmetro para cálculo da FDP Normal.

.desviopd: desvio padrão.

Métodos Públicos:

.zNormal: construtor da classe.

.Gera Amostra: gera uma amostra conforme FDP Normal.

**22) zGeral**

Definição:

Representa a FDP Geral.

Superclasses:

.zFunção

Variáveis:

.nrcoord: número de coordenadas.

.x: array contendo os pontos x.

.y: array contendo os pontos y.

Métodos Públicos:

.zGeral: construtor da classe.

.GeraAmostra: gera uma amostra conforme FDP Geral.

**22) zTWindow**

Definição:

Esta classe é uma classe definida pela Zortech. Ela controla a abertura, edição e fechamento de janelas.

Superclasses: (não possui)

Variáveis:

As variáveis dessa classe são utilizadas apenas por seus métodos, por isso não serão descritas aqui.

Métodos Públicos:

.zTwindow: construtor da classe.

.clear: limpa a janela.

.title: coloca um titulo na janela.

.select: seleciona uma janela.

.locate: posiciona o cursor na linha r e coluna c da janela.

.printf: imprime constantes e/ou valores na janela.

.gets: lê uma string na janela.

.getn: lê um número decimal na janela.

.geti: lê um número inteiro na janela.

.geta: lê um nome com 12 (doze) caracteres na janela.

.getc: lê um caracter na janela.

.close: fecha uma janela.

**24) zMenu**

Definição:

Esta é uma classe definida pela Zortech. Ela controla a escolha de opções em um menu.

Superclasses:

.zScreenArea e zInteractiveObject.

Variáveis:

As variáveis dessa classe são utilizadas apenas por seus métodos, por isso não serão descritas aqui.

Métodos Públicos:

.zMenu: construtor da classe.

.Select: seleciona uma opção do menu.

**25) zDList**

Definição:

Esta é uma classe definida pela zortech. Ela controla funções de uma lista duplamente encadeada.

Superclasses: (não possui)

Classes Amigas:

São necessárias para que suas subclasses possam acessar suas variáveis privadas. São elas:

.zDLCursor (classe do Zortech).

.zListaEventos.

.zFilaPacotes.

.zListaClasses.

.zListaInterfaces.

.zAnelLógico.

Variáveis:

.count: contém o número de elementos existentes na lista, é uma variável privada.

.as demais variáveis não são utilizadas pelo SIMILE, por isso não serão descritas aqui.

Métodos Públicos:

.zDList: contrutor da classe.

.linkin: insere um elemento na lista.

.linkout: exclui um elemento da lista.

.fwd: avança uma posição da lista.

.fwdc: avança uma posição da lista, considera a lista como sendo uma lista circular.

.bkwd: volta uma posição da lista.

.start: posiciona para o início da lista.

.end: posiciona para o fim da lista.

.get: busca um elemento da lista.

Obs: Maiores detalhes sobre as classes zTWindow, zDList e zMenu poderão ser vistos no manual C++ Tools da Zortech [32]. Aqui só foram relacionados as variáveis e os métodos utilizados pelo SIMILE.

# APENDICE C

## MENSAGENS DE ERRO

Durante a entrada dos dados são feitas checagens validando esses dados. De acordo com o tipo de erro ocorrido, são mostradas as seguintes mensagens:

."Nr. de Interfaces Excedeu o Limite": o número de interfaces excedeu o limite estipulado. Esse limite depende da capacidade de armazenamento do computador. Nessa primeira versão do SIMILE, foi previsto um limite de 100 (cem) interfaces, mas esse limite pode ser alterado para um valor variável, através da implementação de uma rotina para cálculo da capacidade de armazenamento do computador.

."Faltou Disc. Liberação Pacote": a tecla <ESC> foi acionada no meio da entrada das informações de uma classe e o usuário informa que não deseja cancelar a execução. Serve apenas para avisar o usuário para informar a disciplina de liberação de pacote.

."Faltou Comprimento Fila": a tecla <ESC> foi acionada no meio da entrada das informações de uma classe e o usuário informa que não deseja cancelar a

execução. Serve apenas para avisar o usuário para informar o comprimento da fila.

."Faltou Disc.Escalonamento": a tecla <ESC> foi acionada no meio da entrada das informações de uma classe e o usuário informa que não deseja cancelar a execução. Serve apenas para avisar o usuário para informar a disciplina de escalonamento de pacotes na fila.

."Faltou Processo Chegada": a tecla <ESC> foi acionada no meio da entrada das informações de uma classe e o usuário informa que não deseja cancelar a execução. Serve apenas para avisar o usuário para informar o processo de chegada.

. "Faltou Tempo de Serviço": a tecla <ESC> foi acionada no meio da entrada das informações de uma classe e o usuário informa que não deseja cancelar a execução. Serve apenas para avisar o usuário para informar o tempo de serviço.

."Valor Maximo Inválido": o valor máximo informado na FDP uniforme é menor do que o valor minimo.

."Nr. Coordenadas Excedeu o Limite": o número de coordenadas informado na FDP normal excedeu o limite permitido. O limite previsto no SIMILE, em sua primeira versão, é igual a 20, podendo ser alterado de acordo com a capacidade de armazenamento do computador.

."Nivel de Prioridade Inválida": o nivel de prioridade informado é diferente de 0, 2, 4, e 6.

."Valor Decimal Inválido": o último valor digitado é um valor decimal inválido.

."Arquivo Inexistente": não existe um arquivo com o nome informado. Essa mensagem é mostrada quando é informado o nome de um arquivo que não existe, que supostamente teriam os parâmetros de entrada do modelo de simulação ou os resultados da simulação.

."Problema na Leitura de Arquivo": não foi possível ler o arquivo informado.

."Arquivo já Existente": já existe um arquivo com o mesmo nome do arquivo informado para gravação dos dados de entrada ou dos resultados da simulação.

."Problema na Gravação de Arquivo": apresentou problema na gravação de um registro do arquivo.

."Nivel de Confiança Inválido": os niveis de confiança permitidos são: 10, 20, 40, 50, 60, 80, 90, 95, 98 e 99.

."Estourou Memória": excedeu o limite de armazenamento da memória do computador.

## REFERÊNCIAS BIBLIOGRAFICAS

[1] ADELSBERGER, HH, Pooch, UW, Shanoon, RE, Williams, GN. *Rule Based Object Oriented Simulation Systems.* In: SIMULATION SERIES, Vol. 17, Nr. 1, 1986. p. 107-112.

[2] BIRTWISTLE, G.M. *Simula begin.* AUERBACH Publisher Inc. Philadelphia, Pa. 1973.

[3] BOOCH, Grady. *Object oriented Design.* The Benjamin/ Cummings Publishing Company, Inc., 1991. 579 p.

[4] BRASILEIRO, Marcos Antonio Gonçalves, MOURA, José Antão Beltrão, CABRAL, Maria Izabel Cavalcanti. *Avaliação de desempenho da integração voz e dados em redes locais com passagem de ficha.* In: 7o. SIMPOSIO BRASILEIRO DE TELECOMUNICAÇÕES, (Setembro 1989). p. 12-19.

[5] BRASILEIRO, Marcos Antonio Gonçalves & MOURA, José Antão Beltrão. *Modelagem de aplicações criticas no tempo em redes locais com passagem de ficha.* In: SEMINAIRE FRANCO-BRASILEIRO SUR LES SYSTEMES INFORMATIQUES REPARTIS. Florianópolis, 1989. p. 257-265.

[6] BRASILEIRO, Marcos Antonio Gonçalves, CABRAL Maria Izabel Cavalcanti, SILVA, Hélio Menezes. *SAVAD - Uma ferramenta para avaliar o desempenho de sistemas distribuídos.* In: SEMINAIRE FRANCO-BRASILEIRO SUR LES SYSTEMES INFORMATIQUES REPARTIS. Florianópolis, 1989. p. 281-288.

[7] BULGREN, W.G. *Discrete system simulation.* Prentice-Hall, In., Englewood Cliffs, N.J., 1982.

[8] CAUPER FILHO, Antonio. *Um simulador para avaliação de desempenho de redes locais.* Tese de Mestrado. Universidade Federal da Paraíba, 1988. 122 p.

[9] COAD, Peter & YOURDON, Edward. *Análise Baseada em Objetos.* Trad. CTI Informática. 2a. Edição. Editora Campus. 1992, 252 p.

[10] DIESCH, Kurt H. & BARTA, Thomas A. *Object-oriented discrete-event simulation in a strongly-typed procedural language.* Object-Oriented Simulation. In: SIMULATION SERIES, Vol. 23, No. 3, 1991. p. 43-49.

[11] DOYLE, Robert J. *Object-oriented simulation programming.* In: OBJECT-ORIENTED SIMULATION, 1990. p. 1-6.

[12] GALDINO, J.F. & GIOZZA, W.F. *Redes Locais de Altíssima Velocidade.* 90. SIMPOSIO BRASILEIRO DE TELECOMUNICAÇÕES. São Paulo, 1991.

[13] GORDON, Geoffrey. *System simulation.* Second edition. IBM Corporation. Prentice-Hall International, INC. 1980. 324 p.

[14] HENDERSON-SELLERS, Brian & EDWARDS, Julian M. *Object-oriented systems life cycle.* Communication of the ACM. Vol. 33, No. 9, 1990. p. 143-159.

[15] IEEE. Project 802. *Local Area Network.* IEEE Computer Society. IEEE 802.3 - CSMA/CD, IEEE 80.4 - Token Bus, IEEE 802.5 - Token Ring, IEEE 802.2 - LLC. 1985.

[16] KLEINROCK, L. *Queueing system.* Vol. 1: Theory, John-Wiley & Sons, 1975.

[17] KREUTZER, W. *System simulation - programming styles and languages.* Addison-Wesley Publishing Company, 1986.

[18] KUEHN, P.J. *Multiqueue systems with nonexhaustive cyclic service.* B.S.T.J. Vol. 58, no. 3, March 1979. p.671-698.

[19] LADD, Scott Robert. *C++ techniques & applications.* M&T Publishing, Inc., 1990. p. 466.

[20] LI, Xining & UNGER, Brian. *Languages for distributed simulation.* In: SIMULATION SERIES, Vol. 18, Nr. 3, 1987. p. 35-39.

[21] MACNAIR, E.A. & SAUER, C.H. *Elements of practical performance modeling.* Prentice-Hall, Inc., 1981.

[22] MOURA, José Antão Beltrão. *Loops and ethernet: Evaluation and comparison of performance and complexity.* Tese de Mestrado. University of Waterloo. 1979. 181 p.

[23] MOURA, José Antão Beltrão et alii. *Redes locais de computadores (protocolos de alto nivel e avaliação de desempenho)*. Editora McGraw-Hill, São Paulo, 1986. 446 p.

[24] PINSON, Lewis J. & WIENER, Richard S. *An introdution to object-oriented programming and smalltalk*. Addison-Wesley Publishing Company, 1988. 501 p.

[25] PRESSMAN, Roger S. *Software engineering*. Second edition, McGraw-Hill International Editions, 1987. 567 p.

[26] SCHRIBER, T.J. *Simulation using GPSS*. John Wiley & Sons. 1974.

[27] SOARES, Luiz Fernando G. *Modelagem e simulação discreta de sistemas*. VII Escola de Computação, São Paulo, 1990. 250 p.

[28] SOARES, Luiz Fernando G. *Redes locais*. Editora Campus. 1990. 250 p.

[29] TACHLITSKY, M, RUGGIERO, WV. *Um Estudo Comparativo por Simulação de Técnicas de Acesso para Redes Locais*. 2o. SIMPOSIO BRASILEIRO DE REDES DE COMPUTADORES. 1984. p. 17-1 a 17-21.

[30] TAROUCO, Liane M.R. *Redes de computadores (locais e de longa distancia)*. McGraw-Hill. 1986. 353 p.

[31] WEINER, Richard S. & PINSON, Lewis J. *Programação orientada para objeto e C++*. Trad. Andréa Nastri. Makron Books do Brasil Editora Ltda. 1991. 333 p.

[32] ZORTECH C++ COMPILER V2.1. *C++ Tools*. 1990, 379 p.