UM SIMULADOR PARA ESTELLE TEMPO-REAL

José Valentim dos Santos Filho

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA

Aprovada por:

Prof. Aloysio de Castro Pinto Pedroza, Dr.

Prof. Jorge Lopes de Souza Leão, Dr.Ing.

Prof. Orlando Bernardo Filho, D.Sc.

Prof. Orlando Gomes Loques Filho, Ph.D.

Prof. Otto Carlos Muniz Bandeira Duarte, Dr.Ing.

RIO DE JANEIRO, RJ - BRASIL

OUTUBRO DE 2002

SANTOS, JOSÉ VALENTIM FILHO Um Simulador para Estelle Tempo-Real [Rio de Janeiro] 2002

XI, 79 p., 29,7 cm, (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2002)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1 – Restrições Temporais 2 – Qualidade de Serviço

3 – Estelle

I. COPPE/UFRJ II. Título (série)

Dedicatória:

Dedico esta Tese ao meu Pai (*In memorian*), a minha Mãe Jerusa (mãinha) e a Alessandra, minha Companheira.

Agradecimentos:

Agradeço, primeiramente a Deus pela dádiva da vida.

Aos meus Pais, por tudo, sempre.

À Alessandra, minha companheira, pelo apoio e incentivo constantes.

Aos Professores do GTA: Aloysio, Leão, Otto e Rezende pela formação técnica aqui recebida.

Agradeço, em particular, ao Prof. Aloysio pelo apoio e incentivos constantes e principalmente pela confiança em mim estabelecida durante todo o tempo.

Ao Prof. Orlando Bernardo que me deu um apoio inestimável durante o desenvolvimento do trabalho.

Aos companheiros do GTA: Renata, Flávio, Paulo André, Rubi, Marcial, Eric, Bernardo, David, Bagatelli, Granato, Vidal, Duboc, Ezequiel, Belem, Aline e em particular: Artur, Saulo, Marcio, Kleber, Pedro, Gardel e Roman.

Aos que dividiram comigo, aqui no Rio, além do aluguel e de todas as outras contas, os sabores e dissabores de uma vida em república: Paulo Sérgio, Álvaro, Moema, Josi, Lene, Cremildo, Magnos, Fagundes e Xéreu (Nivaldo) este último, principal responsável pela minha vinda para o Rio fazer mestrado.

E por fim, ao sofrido povo brasileiro pelo financiamento da bolsa de mestrado e que eu agora presto conta.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UM SIMULADOR PARA ESTELLE TEMPO-REAL

José Valentim dos Santos Filho

Outubro/2002

Orientadores : Aloysio de Castro Pinto Pedroza
: Jorge Lopes de Souza Leão

Programa: Engenharia Elétrica

Esta tese apresenta um simulador para a técnica de descrição formal Estelle Tempo-Real. Esta ferramenta permite a depuração e teste de protocolos especificados em Estelle que contêm restrições temporais fundamentais para aplicações multimídias. Os traços da simulação, bem como gráficos que permitem avaliar o desempenho do protocolo especificado, são fornecidos através de uma interface que deve ser acessada a partir de um *Browser*.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A REAL-TIME ESTELLE SIMULATOR

José Valentim dos Santos Filho

October/2002

Advisors : Aloysio de Castro Pinto Pedroza
: Jorge Lopes de Souza Leão

Department: Electrical Engineering

This thesis presents a Real-Time Estelle Simulator. This tool allows to debug and to test protocols with time constraints. Traces and graphs are provided by the Visual Interface which allows to analisy the performance of the protocol specified. This interface can be accessed using a WEB Browser.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Nos últimos anos as redes de computadores têm ocupado um espaço cada vez maior na nossa sociedade facilitando o acesso a todo tipo de informação. Toda comunicação feita pelos computadores é através de troca de mensagens comandada por protocolos de comunicação que sofrem constantes melhorias. A medida que as tecnologias de rede desenvolvem-se, aumenta a qualidade dos serviços que nos estão sendo disponibilizados, bem como a possibilidade do surgimento de novas aplicações. Mas cada aplicação tem suas características particulares que devem ser tratadas por um protocolo específico.

O projeto de um novo protocolo de comunicação para atender a um determinado tipo de aplicação passa pelo processo de especificação formal, onde o projetista tem a comodidade de projetar uma determinada aplicação distribuída em um nível de abstração maior do que o obtido com o uso de uma linguagem de programação, que envolve detalhes muito específicos relacionados ao ambiente de implementação.

Neste contexto, as Técnicas de Descrição Formal (TDFs) assumem um papel importante no projeto de protocolos. As técnicas mais tradicionais e utilizadas são Estelle [1] e LOTOS [2].

A linguagem Estelle é uma TDF padronizada pela ISO (International Standards Organization). O uso de Estelle para especificação, validação e implementação semi-automática de protocolos tradicionais e de sistemas distribuídos tem sido realizado com sucesso como se pode comprovar pela divulgação de algumas ferramentas de projeto comerciais [3] e acadêmicas [4].

Na última década houve um excepcional aumento do poder de processamento dos computadores. Este fator aliado ao crescimento das redes de alta velocidade tornaram possíveis o surgimento de novos tipos de serviços que pudessem usar ao máximo a tecnologia hoje existente. Aplicações como transmissão de voz e de vídeo em redes de computadores encontram bastante utilidade em diversas áreas. Vídeo entretenimento, telefonia sobre IP (*Internet Protocol*), videoconferência, jogos interativos, ensino à distância entre outras, mas este tipo de aplicação exige que determinadas restrições de tempo sejam atendidas, obrigando que os novos protocolos dêem um tratamento adequado a estes requisitos para atingir a Qualidade de Serviço (QoS) desejada.

Um outro exemplo, a ser citado, de protocolos com restrições temporais é o de controle de *handoff* para usuários móveis. O *handoff* é um processo que ocorre quando um usuário deixa uma célula na qual ele está se comunicando com uma estação base e entra em uma outra célula, com outra estação base sendo responsável pela sua comunicação. É necessário que se estabeleça um novo caminho entre o usuário móvel e a nova estação. A limitação de tempo neste processo é importante para que não haja falha na comunicação.

Apesar de sua ampla utilização para especificar protocolos tradicionais, a padronização Estelle tem uma importante limitação no que diz respeito a especificação formal de protocolos que envolvam restrições temporais fim-a-fim. A única cláusula de Estelle que permite atribuir alguma restrição de tempo é a cláusula *delay*, mas ela somente atribui a restrição a uma dada transição, considerada de maneira isolada. Sendo portanto, incapaz de impor uma restrição que envolvesse mais de uma entidade do sistema.

Assim como Estelle, a forma padrão da linguagem Lotos também não contempla restrições de tempo consideradas fim a fim. Desta limitação das técnicas tradicionais surgiu a necessidade de desenvolver extensões que permitissem aos usuários fazer especificação formal de protocolos para aplicações multimídias, tal como RT-LOTOS [5]. A linguagem *Real-Time* Estelle [6] é o resultado de uma extensão desenvolvida para Estelle tornando-a capaz de especificar restrições temporais fim a fim para sistemas distribuídos.

Esta extensão foi construída visando seguir a mesma sintaxe da Estelle padrão incorporando novos operadores e palavras-chave à linguagem. A descrição das restrições é feita através de fórmulas de lógica temporal.

Desta extensão utilizamos um subconjunto que nos permitisse especificar requisitos de tempo em protocolos de aplicações multimídias. Os parâmetros mais comuns de QoS para este tipo de aplicação encontrados na literatura são: vazão, atraso,*jitter*(variações de atraso) e taxa de erros.

Neste tipo de aplicação as restrições temporais são tão, ou mais importantes, que os aspectos funcionais.

No sentido de fornecer ao usuário de Estelle uma ferramenta de fácil uso para que ele possa especificar, compilar e simular suas especificações de protocolos, que envolvem restrições temporais fim-a-fim, com o intuito de depurar possíveis erros de projeto, foi desenvolvido o sistema integrado de auxílio ao projeto de protocolos de comunicação com restrições temporais - ProtCAD/RT [7].

O sistema conta com uma Interface Visual, na qual a especificação em *Estelle Tempo-Real* se dá de maneira bastante simples e intuitiva, tornando mais rápido e fácil o processo de especificação.

A interface se encarrega de, a partir da representação visual, gerar a especificação textual que servirá de entrada para o compilador. Esta é uma enorme facilidade pois não obriga o usuário a ter um profundo conhecimento da sintaxe de Estelle para especificar um protocolo.

A partir da especificação textual em Estelle o compilador gera as formas intermediárias que armazenam as informações estáticas, ou seja, com relação a topologia da simulação, as informações dinâmicas que armazenam os procedimentos que serão utilizados pelo simulador durante o processo de simulação e as informações sobre as restrições temporais impostas ao sistema.

Neste trabalho será apresentado um simulador para Estelle Tempo-Real capaz de tratar as restrições temporais impostas na especificação. O simulador monitora a interação desde o momento de sua transmissão até a chegada ao receptor, neste momento é feito o cálculo do atraso associado a cada interação. São realizados também o cálculo do *jitter* máximo, da vazão média e da taxa de erros. O simulador mostra ainda os traços da simulação com intuito de detectar alguma situação de *deadlock*.

A interface do simulador foi desenvolvida em Java por causa da sua independência de plataforma, podendo ser utilizada através da WEB, não precisando, portanto, que o usuário tenha o sistema instalado na sua própria máquina.

O capítulo 2 aborda a sintaxe da Estelle Tempo-Real, fazendo uma breve descrição do comportamento das palavras-chave e dos novos operadores acrescentados à linguagem Estelle Padrão, bem como, alguns exemplos de especificação dos parâmetros relacionados à Qualidade de Serviço

O capítulo 3 faz uma descrição do ProtCAD/RT como um todo, abordando o relacionamento entre todos os módulos do sistema. Trata principalmente da implementação do simulador, detalhando o tratamento dado à questão temporal, o mecanismo utilizado para monitorar as interações, e como são feitos os cálculos dos parâmetros especificados. A construção da interface gráfica e a comunicação entre o módulo cliente e o módulo servidor também serão descritos nesse capítulo.

No quarto capítulo, são feitos alguns exemplos de especificação em Estelle Tempo-Real, mostrando os traços da simulação, os gráficos de *delays* especificados, *jitter* máximo encontrado, cálculo da vazão média, vazão instantânea e taxa de erros. No último capítulo são apresentadas as conclusões sobre o trabalho e sugestões para trabalhos futuros.

Capítulo 2

A Extensão Estelle Tempo-Real

2.1 Introdução

As técnicas de descrição formal têm sido úteis na especificação de protocolos e de sistemas distribuídos tradicionais. Infelizmente, a maioria dessas técnicas, ou linguagens, não contemplam o conceito de tempo real, tornando impossível a descrição de sistemas multimídias distribuídos que fazem restrições em termos de taxa de transferência, retardo na transmissão, tempo de resposta, etc. Estes requisitos são usualmente expressos através de parâmetros de Qualidade de Serviço (*QoS*). Este capítulo trata da Estelle Tempo-Real, extensão criada que visa preencher uma lacuna deixada pela técnica na sua forma padrão. A seção 2.2 aborda os parâmetros de qualidade de serviço mais importantes, segundo a literatura existente, que podem ser especificadas em Estelle Tempo-Real. A seção 2.3 explica com detalhes a limitação da Estelle padrão, e a seção 2.4 descreve os operadores e as palavras-chave introduzidas na extensão utilizadas para especificar as restrições temporais fim-a-fim. Alguns exemplos de especificação de restrições temporais são encontrados na seção 2.5 e na seção 2.6 os comentários finais deste capítulo.

2.2 Parâmetros de *QoS*

As aplicações multimídias, que são o foco desse estudo, têm requisitos de tempo real expressos através de parâmetros de Qualidade de Serviço. Os parâmetros mais comuns encontrados são os seguintes: atraso, *jitter*(variações de atraso), vazão e taxa de perdas.

Mídias como voz e vídeo são bastante sensíveis ao atraso, ou *delay*. Ele é definido como sendo a soma dos retardos de acesso e do retardo de transmissão. O retardo de acesso é o intervalo de tempo decorrido desde que uma mensagem a transmitir é gerada pela estação até o momento em que a estação consegue transmitir esta mensagem com sucesso e o retardo de transmissão é o intervalo de tempo decorrido desde o início da transmissão de uma mensagem por uma estação de origem até o momento em que a mensagem chega à de destino [8]. O atraso indica o nível de congestionamento da rede e constitui uma restrição importante para tais mídias. Em uma conversa telefônica, um atraso de até 150ms não é perceptível ao ouvido humano, entre 150 e 400ms não é o ideal, mas pode ser aceitável e acima de 400ms causa sérios danos a interatividade [9]. O atraso depende fundamentalmente da distância entre as entidades que estão se comunicando, da capacidade do enlace de comunicação e do poder de processamento dos roteadores ao longo da rede.

Em mídias contínuas que exigem cadência na transmissão, a violação da restrição *jitter*, que é a variação do atraso entre os pacotes, provocaria instantes de silêncio causada pela não uniformidade na recepção das mensagens.

A vazão é definida como sendo a taxa de transmissão líquida, ou seja, é a diferença entre o que foi transmitido e o que foi perdido durante a transmissão, e cada tipo de mídia tem seus requisitos de vazão específicos.

As mídias contínuas, apesar da severidade com relação as restrições de atraso e *jitter*, se mostram tolerantes com relação a eventuais perdas de pacotes durante a transmissão. Dependendo de como a mídia é codificada, transmitida e tratada no receptor, a tolerância pode chegar a até 20% de perdas dos pacotes. Porém, na especificação de restrições como atraso e vazão, é preciso que haja um controle fim-a-fim de mensagens, envolvendo o disparo de mais de uma transição e podendo envolver mais de um módulo. Na Estelle padrão a especificação de restrições temporais fim-a-fim se torna inviável. A seção seguinte aborda com detalhes a limitação da Estelle padrão.

2.3 Limitação da Estelle Padrão

No processo de especificação formal de protocolos, o projetista deve tentar retratar o comportamento mais próximo do sistema alvo, utilizando os recursos oferecidos pela técnica em questão. A cláusula *Delay* é o único recurso oferecido por Estelle que permite impor uma restrição temporal. No entanto, essa restrição é atribuída a uma única transição, considerada de maneira isolada. Quando se trata da especificação de protocolos que envolvem restrições temporais fim-a-fim, a Estelle padrão se mostra inapropriada [10].

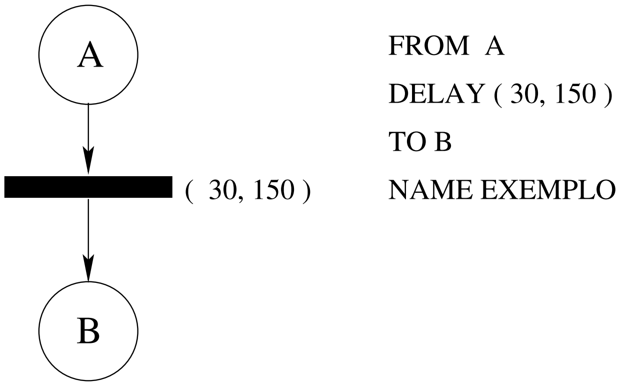

Figura 2.1: Especificação de cláusula *Delay*.

A figura 2.1 mostra uma simples transição que está sofrendo uma restrição temporal através da cláusula *Delay*, no exemplo a transição não irá disparar até 30 unidades de tempo, irá disparar com uma determinada probabilidade entre 30 e 150 unidades de tempo, e certamente irá disparar depois de atingir 150 unidades de tempo, seguindo a semântica dos sistemas de transições temporizadas [11]. No entanto, esta restrição vale somente para a transição citada. Não há, na Estelle padrão, como especificar restrições temporais que envolvam mais de uma transição do sistema. Desta limitação surgiu a necessidade de utilizar uma extensão à Estelle denominada *Real-Time Estelle* que permite fazer a especificação de parâmetros de tempo que envolvesse mais de uma entidade do sistema. Essa extensão teve sua sintaxe construída visando manter a mesma estrutura da Estelle Padrão. A abordagem utilizada na definição desta linguagem, bem como os operadores temporais incorporados serão encontrados na seção seguinte.

2.4 Sistemas de Tempo-Real

Pode-se definir a semântica formal para sistemas de tempo-real como sendo um conjunto de seqüências de estados temporizados, onde cada estado representa uma possível situação do sistema. A lógica temporal tradicional provê uma maneira suscinta e natural de expressar requisitos qualitativos de tempo de sistemas. No entanto, os operadores temporais tradicionais são insuficientes para a especificação de requisitos quantitativos de tempo que impõem limites no comportamento de sistemas reativos [12].

A *Real-Time* Estelle seguiu a mesma abordagem utilizada no desenvolvimento da *Quality of Service Temporal Logic* (QTL) [13] e da extensão para a *Specification Description Language* (SDL) [14], onde foi criada uma linguagem híbrida, cujo o comportamento funcional do sistema é descrito por automata ou fórmulas algébricas, enquanto que as restrições temporais são feitas com lógica temporal. Então o sistema tempo-real é caracterizado por todas as seqüências de estados temporizados produzidos pelo automata ou algebra em adição às fórmulas de lógica temporal. A *Real-Time* Estelle teve sua sintaxe desenvolvida de maneira que fosse flexível o suficiente para ter uma semântica descritiva e operacional.

A parte operacional foi extraida da Estelle padrão e a parte descritiva foi incorporada pela extensão para tempo real. A especificação formal de um sistema com restrições temporais em Estelle Tempo-Real consiste, portanto, de uma seqüência de observações temporizadas cujas as seqüências de estados temporizados são construídas usando a semântica operacional da Estelle padrão e da extensão para tempo real, descrita pela *Real-Time* Estelle. Em um passo seguinte as seqüências são usadas como modelos para as fórmulas de lógica temporal.

Neste trabalho foi utilizado apenas um subconjunto de operadores temporais capazes de especificar restrições temporais de aplicações multimídias conforme exemplos verificados na literatura [15]. Este conjunto de acréscimos à TDF Estelle é mostrado na tabela 2.1. Informações mais detalhadas sobre estes e outros operadores podem ser encontradas em [16].

Neste formalismo específico, a descrição funcional do sistema, ou seja, como ele progride em relação às interações possíveis, utiliza a mesma estrutura de uma especificação em Estelle de protocolos tradicionais sem restrições temporais. Entretanto, no domínio dos requisitos de QoS desejados é que se emprega a extensão, utilizando uma seção denominada *time constraints*, onde se descrevem as restrições específicas daquele subsistema em questão. Desta maneira, separa-se a análise de correção do comportamento do sistema da análise de seu desempenho, enquanto especificação de sistema com requisitos temporais.

A partir da tabela 2.1 pode-se descrever a interpretação dos operadores lógicos e temporais citados, bem como das palavras-chave e descritores de estado. A palavra-chave *time constraints* indica o início de um bloco de declaração de restrições temporais referentes ao corpo de módulo onde se encontra. Restrições fim-a-fim devem ser descritas em um corpo de módulo pai que engloba as entidades comunicantes, com isto, preservam-se as regras de escopo presentes na TDF Estelle.

Tabela 2.1: Acréscimos à TDF Estelle Padrão

Tipo de Identificador	Identificadores
Palavras-Chave	*time constraints*
Operadores Lógicos	*and* *or* *otherwise* *eventually* *henceforth* *leadsto* *forbids* *at*
Operadores Temporais	*now*
Descritores Simples de Estado	*sending of* *receiving of*

O operador temporal *now* tem funcionamento semelhante a uma função que busca a hora atual de um sistema. Ou seja, ele devolve o instante de tempo no qual o sistema se encontra, seja ele simulado ou encontrado nas funções de hora do sistema alvo em questão.

Já o operador *otherwise* deve ser usado para executar uma nova ação, caso a avaliação de uma restrição específica não tenha sido satisfeita. Isto permite ao projetista planejar um novo comportamento para o protocolo.

Os descritores simples de estado utilizados verificam o conteúdo dos pontos de interação do sistema em execução. O descritor *sending of* indica a saída de uma primitiva por um determinado ponto de interação, e *receiving of* a chegada de uma mensagem em um outro ponto de interação. Nestes descritores também são identificados o módulo-filho ao qual a primitiva se refere, seguido do ponto de interação. Além de permitir o acréscimo de números de seqüência que indicam a ordem das mensagens enviadas.

O operador unário *eventually* indica que um determinado evento irá ocorrer em algum instante de tempo, a partir do instante de observação do sistema. Já o outro operador unário *henceforth* indica que um dado evento sempre ocorrerá, também a partir do instante de observação do sistema.

Os outros operadores lógicos podem ser definidos com base nos elementos descritos anteriormente e em operadores comumente utilizados com a interpretação esperada, como mostram os exemplos a seguir. Considerando as seguintes equações, onde a e b são fórmulas e t uma variável temporal:

1. (a `leadsto` b) $\equiv$ (a `implies` `eventually` b)

2. (a `forbids` b) $\equiv$ (a `implies` `henceforth(` `not` b))

3. (a `at` t) $\equiv$ (a `and` ($t = now$))

Esta é a semântica de uma lógica de primeira ordem com variáveis de tempo. Ela é bastante similar a *Real-Time Temporal Logic* (RTTL) [17]. Como resultado, a *Real-Time* Estelle é muito expressiva, mas indecidível. Outras semânticas seriam sufientes para muitas aplicações como a *Metric Temporal Logic* (MTL), que utiliza uma lógica decidível, mas existem diversos exemplos que não seriam especificáveis com MTL. O principal motivo pelo qual esta semântica foi escolhida é a sua expressividade.

Um exemplo de especificação de uma restrição temporal é dado a seguir: um módulo tem algumas transições que passarão do estado *IDLE* para o estado *CONNECTED*. Esta transição deve ser executada dentro de no máximo 5 unidades de tempo após a sua ativação. O trecho em Estelle Tempo-Real seria da seguinte forma:

```
TRANS
    FROM IDLE TO open_conection begin
output mcep[0].DATREQ;
   end;

    FROM open_conection TO CONNECTED
        WHEN MCEP[0].CONFIRMATION begin end;
```

```
A parte com as restrições temporais
encontra-se na seção TIME CONSTRAINTS:

TIME CONSTRAINTS
    NAME tc1:
    FORALL t:time;
        HENCEFORTH (
        MAJOR_STATE(IDLE) AT t LEADSTO
            MAJOR_STATE(CONNECTED) AT (now<=t+5));
```

No exemplo acima, a restrição imposta diz que sempre que o estado *IDLE* estiver ativo, o estado *CONNECTED* deve ser alcançado em, no máximo, 5 unidades de tempo. Agora que são conhecidos os operadores lógicos utilizados, vamos exemplificar alguns tipos de restrição.

2.5 Exemplos de Especificação de Parâmetros com Restrições Temporais

Agora tendo como base a figura 2.2, que mostra uma arquitetura simples de um modelo produtor-consumidor, que aqui será de uso didático, serão descritos alguns exemplos de especificação de restrições temporais. O produtor gera pacotes a uma taxa aleatória, o meio é modelado sem perdas, simplesmente dá um atraso também aleatório e o consumidor apenas retira o pacote do meio.

2.5.1 Especificação de Taxa de Transferência

Cada mídia tem seu tipo de tráfego característico, voz e vídeo se caracterizam por rajadas de tráfego, no entanto, é bastante complicado tentar reproduzir fielmente tal característica em uma especificação Estelle. A especificação de uma taxa de transferência será o primeiro exemplo:

```
TIME CONSTRAINTS
```

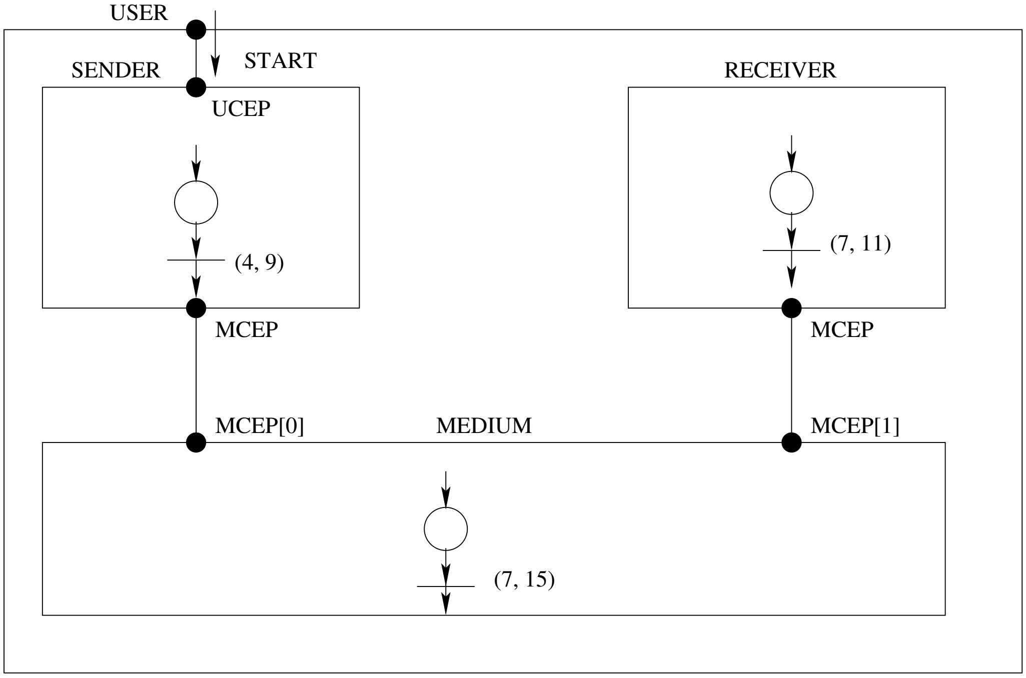

Figura 2.2: Arquitetura de um modelo produtor-consumidor.

```
NAME minima-taxa-de-transferência;
FORALL t : time;
HENCEFORTH(
   SENDING_OF MCEP[0].M_DATReq[z] AT t LEADSTO
   SENDING_OF MCEP[0].M_DATReq[z+1]  AT (t < now <= t+5));
```

Tal restrição indica que, sempre que a primitiva DATAReq for enviada através do ponto de interação MCEP[0] em um tempo t, outra instância da primitiva será enviada no mesmo ponto, passados no máximo, 5 unidades de tempo. O operador SENDING_OF caracteriza a ação de enviar a primitiva. É importante destacar que a taxa de transferência está intimamente ligada com o tamanho da mensagem que está sendo transmitida, tamanho esse que é fixo. A restrição imposta à taxa de transferência não significa que esta taxa seja exatamente a taxa de recepção, pois a transmissão está sujeita a perdas no meio, caso este tenha sido modelado desta forma.

2.5.2 Especificação de Vazão

A vazão efetiva deve ser especificada com relação ao receptor, já prevendo eventuais perdas. A vazão é a taxa de transferência menos a taxa de perda de pacotes, ela depende da capacidade de processamento e dos *buffers* nos roteadores. A vazão também depende do tamanho das mensagens que estão sendo transmitidas.

```
TIME CONSTRAINTS

NAME vazao-efetiva;
FORALL t : time;
HENCEFORTH(
   RECEIVING_OF MCEP[0].DATInd[z] AT t LEADSTO
    RECEIVING_OF MCEP[0].DATInd[z+1]  AT (t < now <= t+10));
```

Aqui, sempre que chegar uma primitiva DATInd pelo ponto MCEP[0] em um tempo t, outra primitiva DATInd deve chegar no mesmo ponto em no máximo 10 unidades de tempo. O operador RECEIVING_OF caracteriza o recebimento da primitiva.

2.5.3 Especificação de Atraso

Dois fatores introduzem retardo na transmissão, o tempo de processamento local e o tempo que um pacote permanece no meio de comunicação. A especificação de uma restrição *delay* poderia ser a seguinte:

```
NAME delay-maximo-meio:
FORALL t : time;
FORALL z : integer;

HENCEFORTH(
SENDING_OF MCEP[0].M_DATReq[z] AT t LEADSTO
RECEIVING_OF MCEP[1].DATInd[z] AT ( now <= t + 7)
```

A restrição acima diz que, sempre que uma interação DATAReq for enviada pelo ponto de interação MCEP[0], a interação DATAInd deverá ser recebida no ponto de interação MCEP[1] em no máximo 7 unidades de tempo depois do envio. Caracterizando portanto o tempo máximo que a interação deve permanecer no meio de comunicação.

Outro ponto importante a ser considerado pelo projetista é a quantidade de tempo que será gasta pelo módulo processador do sistema hospedeiro para o tratamento de um determinado tipo de tráfego. O usuário deve ter a sensibilidade e conhecimento da capacidade do sistema alvo para estimar o intervalo de tempo que será colocado na cláusula *DELAY* de Estelle. Esse tipo de restrição poderia ser feita da seguinte maneira:

```
NAME tempo-de-processamento:
FORALL t : time;

HENCEFORTH (
RECEIVING_OF UCEP.START AT t LEADSTO
SENDING_OF MCEP[0].DATAReq AT ( now < t+10));
```

Nesse caso, quando a interação $START$ é recebida no ponto de interação $UCEP$,dando início ao processo de transferência de dados, começa então o envio das interações $DATReq$ através do ponto de interação $MCEP[0]$ em no máximo 10 unidades de tempo. Impondo um tempo máximo de processamento a uma camada do protocolo.

Em protocolos orientados a conexão, há a fase de estabelecimento de conexão antes que os dados comecem a ser transmitidos. Principalmente em sistemas de tempo real, é importante limitar o tempo para que esta etapa seja cumprida.

```
NAME tempo-estabelecimento-conexão;
FORALL t : time;
HENCEFORTH  (
SENDING_OF MCEP[0].CONNReq AT t LEADSTO
(RECEPTION OF MCEP[0].CONNcnf OR
RECEPTION OF MCEP[0].DISInd) AT (now <= t + 8));
```

Essa restrição diz que, sempre que a interação $CONNReq$ for enviada pelo ponto de interação MCEP[0], solicitando o estabelecimento de uma conexão, deverá haver a recepção no mesmo ponto de interação em no máximo 8 unidades de tempo da primitiva $CONNcnf$, que confirma a conexão, ou da primitiva $DISInd$ que nega a conexão.

2.5.4 Especificação de *jitter*

O *Jitter* ou variação de atraso entre os pacotes é fundamental em aplicações multimídias que exigem cadência. A transmissão destas mídias deve ser cadenciada, ou seja, os pacotes devem ser enviados em intervalos de tempo iguais. Como segue:

```
NAME envio-cadenciado
FORALL t : time;
 HENCEFORTH (
(SENDING_OF MCEP[0].DATReq AT t FORBIDS
    SENDING_OF MCEP[0].DATReq AT (t < now < t+6 )) AND
(SENDING_OF MCEP[0].DATReq AT t LEADSTO
SENDNG_OF MCEP[0].DATReq AT (now = t+7)));
```

A restrição garante que sempre que for feito o envio da interação $DATReq$, pelo ponto de interação $MCEP[0]$ em um tempo t, outra interação $DATReq$ deve ser enviada, no mesmo ponto, exatamente 7 unidades de tempo depois do envio da primeira.

Essa restrição só diz respeito ao lado emissor e poderia ser imposto na especificação através da cláusula *DELAY* de Estelle, atribuindo um intervalo de disparo tal como (5, 5).

Mas somente exigir que o envio seja cadenciado não garante que os pacotes vão chegar em intervalos iguais, pois os pacotes ainda estão sujeitos as condições de congestionamento do meio de transmissão, podendo permanecer muito tempo nas filas dos roteadores. Portanto deve-se especificar o *jitter* de maneira global para garantir que os pacotes chegaram com atrasos dentro de uma faixa pré-estabelecida.

```
NAME jitter-global:
FORALL t : time;
HENCEFORTH (
(SENDING_OF MCEP[0].DATReq AT  FORBIDS
       RECEIVING_OF DATInd[].T_DATInd AT (now < t + 4)) AND
(SENDING_OF MCEP[0].DATReq AT t LEADSTO
       RECEIVING_OF MCEP[1].T_DATInd AT ( now <= t + 7)));
```

Dessa maneira, impõe-se uma variação no atraso de cada pacote entre 4 e 7 unidades de tempo após o envio. O exemplo acima também poderia ter sido descrito de uma outra maneira, mais simplificada.

```
SENDING_OF MCEP[0].DATReq AT t LEADSTO
RECEIVING_OF MCEP[1].DATInd AT (t + 4 <= now <= t+7 )
```

Uma técnica bastante utilizada na tentativa de eliminar, ou pelo menos minimizar o *jitter* é armazenar os primeiros pacotes em um *buffer* no receptor durante algum tempo, este tempo não pode violar a restrição de *delay* e entregá-los cadenciadamente à aplicação.

2.6 Conclusões

Este capítulo comentou os principais parâmetros de QoS para aplicações multimídia, destacando a limitação da Estelle padrão em especificar esse tipo de requisito. Abordou também a Estelle Tempo-Real fazendo uma descrição dos operadores lógicos utilizados e de como a extensão permite especificar restrições temporais fim-a-fim. Também são mostrados exemplos de especificação de alguns tipos de parâmetros. No capítulo seguinte será descrito o conjunto de ferramentas ProtCAD/RT e como ela é utilizada para especificar protocolos com restrições temporais, em particular, será enfatizado o simulador, definindo a FI Tempo-Real, detalhando aspectos de implementação e o mecanismo criado para monitorar as mensagens fim-a-fim. Será descrita também a Interface desenvolvida para utilizar o simulador destacando suas principais características. Detalhes da comunicação Cliente/Servidor também serão abordados.

Capítulo 3

Simulador para Estelle Tempo-Real

3.1 Introdução

As ferramentas de auxílio ao projeto de protocolos são fundamentais para que projetistas possam fazer uma análise comportamental e do desempenho dos sistemas especificados. Este capítulo apresenta um simulador para Estelle Tempo-Real, capaz de simular especificações Estelle com restrições temporais fim-a-fim. O simulador faz parte de um conjunto de ferramentas de uso didático chamado ProtCAD/RT que será abordado na seção 3.2. A seção 3.5 trata dos detalhes de implementação, o mecanismo criado par monitorar mensagens e os resultados gerados. A interface com o usuário, comentando suas principais características, é mostrada na seção 3.6 e os detalhes da comunicação entre cliente e servidor são discutidos na seção 3.7. Ao final, na seção 3.9 são feitos alguns comentários sobre a ferramenta.

3.2 ProtCAD/RT

O Sistema Integrado de Auxílio ao Projeto de Protocolos com Restrição Temporal - PROTCAD/RT surgiu da idéia de ter um conjunto de ferramentas que permitisse ao usuário especificar, compilar e simular especificações de protocolos com restrições temporais fim-a-fim a partir de uma interface visual.

Esta ferramenta permite que projetistas possam utilizá-la para seus estudos em projetos de protocolos e de sistemas distribuídos em geral utilizando Estelle como técnica de descrição formal.

O sistema conta com uma interface visual, um compilador e um simulador, mas outras ferramentas podem ser desenvolvidas e integradas ao sistema como um gerador de seqüências de testes, um verificador e um implementador.

A ferramenta trabalha de maneira integrada, conforme a arquitetura mostrada na figura 3.1. O usuário a partir da interface visual gera o código Estelle Tempo-Real que será compilado gerando a forma intermediária, esta última é utilizada pelo simulador. Tanto a Interface Visual como o Simulador provêem uma interface com o usuário.

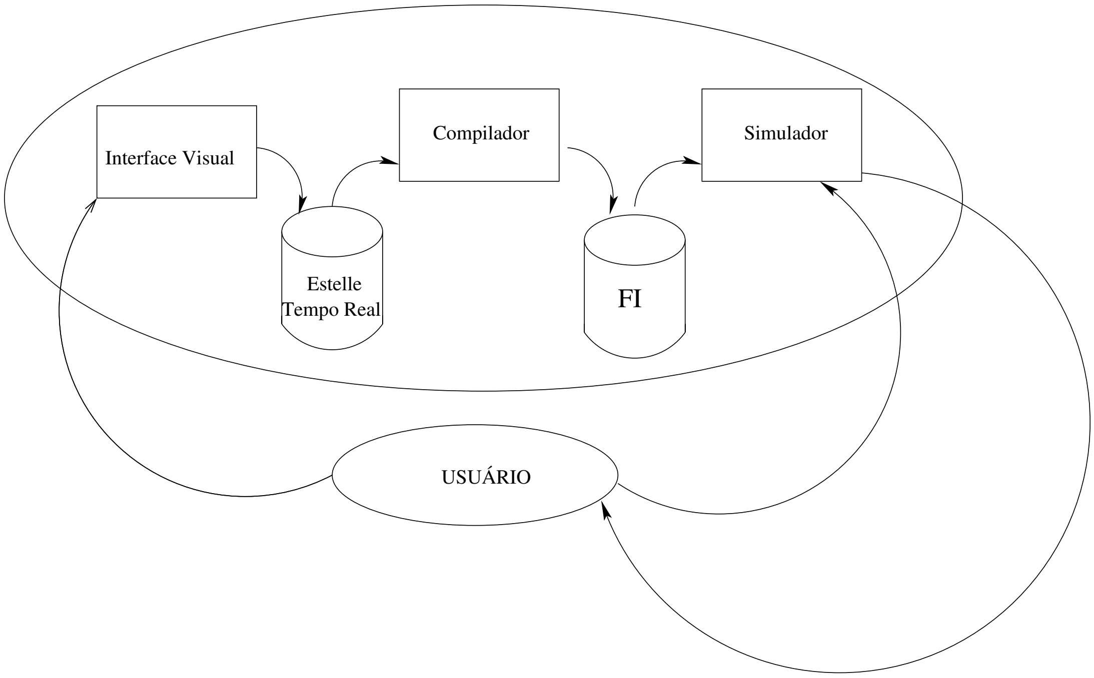

Figura 3.1: Arquitetura do ProtCAD/RT

3.2.1 A Interface Visual

O principal objetivo da Interface Visual [18] é auxiliar o usuário a especificar protocolos de comunicação (que contenham ou não restrições temporais) de maneira organizada, intuitiva e com maior comodidade em relação a codificação textual direta.

Esta interface está capacitada a gerar, sempre que solicitada, o código textual contendo a especificação do protocolo utilizando uma técnica de descrição formal (TDF) adequada (para esta proposta foi escolhida a TDF Estelle). A base desta proposta é o mapeamento dos componentes da linguagem Estelle em objetos que possuam uma representação visual.

Um dos diferenciais desta proposta em relação a outras existentes é a escolha de um ambiente de implementação que permita a sua utilização através da INTERNET e a independência de plataforma, o que levou a escolha da linguagem JAVA.

3.2.2 Compilador para Estelle Tempo-Real

O Compilador para protocolos descritos em Estelle Tempo-Real tem por finalidade gerar a forma intermediária, dividida em uma parte estática, chamada FI Estática e outra dinâmica, chamada FI Dinâmica, que servem de entrada para o simulador e, eventualmente, para outros módulos do sistema que venham a ser desenvolvidos. Estas formas seguem o padrão definido em Oliveira [19] e por Delfino et al [7]. Além disso uma terceira parte referente as questões temporais foi definida, sendo chamada de FI Tempo-Real. Portanto a especificação em Estelle Tempo-Real de um determinado protocolo gera três FIs que serão descritas a seguir:

i) *FI Estática:* Contém os dados referentes a topologia da especificação, tais como o sistema de transições do protocolo, seus estados, seus módulos constituintes, condições de habilitação das transições, entre outros. Em resumo, trata da formalização textual das regras do sistema de transições do protocolo, bem como da estrutura hierárquica da especificação.

ii) *FI Dinâmica:* Contém as informações a respeito do protocolo, como as ações efetuadas durante o disparo de uma transição. Além disto, são introduzidas as funções e procedimentos em Pascal referentes à manipulação dos dados de sistemas e subsistemas referidos na especificação. Serão incorporados também a esta FI os procedimentos de avaliação das restrições temporais impostas.

iii) *FI Tempo-Real:* Contém informações a respeito das restrições temporais, quais restrições estão sendo impostas, sobre quais eventos e interações, os pontos do sistema em que as restrições devem ser observadas.

3.3 Simulador para Estelle Tempo-Real

O simulador para Estelle Tempo-Real oferece ao projetista um suporte para o teste de protocolos especificados em Estelle com restrições temporais com o objetivo de detectar condições de mau funcionamento dos protocolos, além disso, fornecer ao usuário medidas que permitam avaliar o desempenho do protocolo especificado.

Neste simulador foi utilizada a técnica de validação por simulação randômica de WEST [20], nesta técnica apenas uma transição, escolhida aleatoriamente, é analisada a partir do estado corrente ao invés da exploração sistemática de todas as transições.

A técnica consiste na exploração contínua de seqüências de estados da árvore de alcançabilidade. A simulação randômica não necessita de uma base de dados para armazenar os estados já percorridos, evitando assim, o problema da explosão do espaço de estados. Todavia, há duas desvantagens no uso desta técnica: primeiramente, não há uma terminação clara, pois não existe forma de determinar se todos os estados do sistema foram explorados; e mesmo que se defina um critério de parada, não existe garantia de que o sistema foi totalmente verificado [21].

Portanto, o simulador não vale como prova formal de que o protocolo em questão irá funcionar perfeitamente.

Como já foi dito, a integração do simulador com o compilador para Estelle Tempo-Real é feita através das formas intermediárias estática, dinâmica e tempo-real, que servirão como entrada para o simulador, veja a figura 3.2.

A partir da FI estática ele cria objetos que armazenam todas as informações relacionadas à topologia da especificação. Da FI dinâmica são usados os procedimentos criados que serão solicitados ao longo do processo de simulação, e da FI Tempo-Real ele é capaz de identificar todos os tipos de restrições temporais que estão sendo impostas e quais as interações que estão sofrendo tais restrições.

3.3.1 Motor do Simulador

O motor do simulador executa o modelo de máquina de estados finita, monitorando a trajetória das interações fim-a-fim. Ele cria filas (FIFO) interconectando as diversas Máquinas de Estados Finitas, manipula as estruturas particulares ao ambiente de simulação que são, entre outras, o conjunto de transições disparáveis e os estados dos módulos. Os módulos do simulador se relacionam segundo uma regra hierárquica definida por Valim [22] e o algoritmo básico do motor foi baseado no padrão definido em Loureiro [23] e é descrito a seguir:

i) criação de uma lista contendo todas as transições habilitadas;

ii) escolha aleatória de uma transição de entrada desta lista a ser disparada;

iii) execução da ação correspondente;

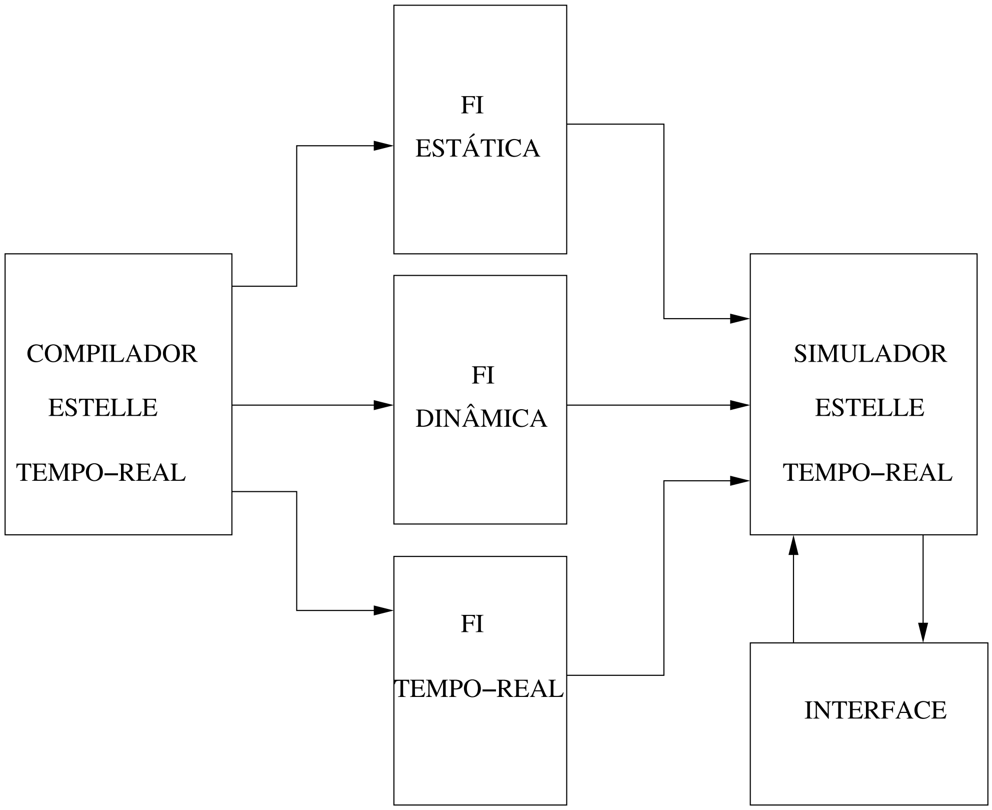

Figura 3.2: Diagrama de Blocos do Simulador

iv) teste do ponto de parada. Se o teste é verdadeiro, os traços de simulação são mostrados na Interface, senão, retorna ao item (ii).

O simulador oferece ao usuário as seguintes facilidades:

- acesso ao simulador através de um *Browser*
- obtenção de traços de simulação;
- obtenção de gráficos de *Delay, Vazão*
- acesso às medidas dos requisitos de tempo-real: *Delay*, Taxa de Erros, Vazão Média e *Jitter*.

3.4 Tratamento dos Requisitos de Tempo-Real

A cláusula *Delay* de Estelle permite que se especifique restrições de tempo para disparo de uma transição, mas considerada de maneira isolada. Isto impõe uma limitação à linguagem de não poder especificar restrições temporais entre dois ou mais módulos.

A extensão Estelle Tempo-Real veio permitir que esta limitação fosse superada, permitindo que restrições fim-a-fim fossem especificadas, cabendo ao simulador, a missão de monitorar e computar os valores de tempo das mensagens, fornecendo informações ao usuário para que este se certifique de que as restrições foram atendidas. É através da FI Tempo-Real que o simulador obtém as informações sobre as restrições temporais que estão impostas. Esta forma intermediária será detalhada na seguinte seção.

3.4.1 A FI Tempo-Real

É dado a seguir um exemplo de FI Tempo-Real para um modelo que impõe restrições de *Delay* e Vazão. Toda a especificação ESTELLE deste modelo pode ser encontrado no apêndice A.

```
#TempoReal.in --- arquivo que armazena as informações
#referentes as restrições temporais
[delay]
[d]
corpodopai SENDRECVBODY
cabecalho   MEDIUMTYPE
ipdeentrada MCEP[0]
ipdesaida MCEP[1]
corpodocanal  SENDERRECEIVER
interacaodeentrada  DATREQ
interacaodesaida  DATIND
[fd]
[fim_delay]

[vazao]
tempodemedida 2000
tamanhodopacote 1500
tempoderajada 50
```

```
paidomeio  SENDRECVBODY
cabecalhodomeio MEDIUMTYPE
ipdeentrada MCEP[0]
ipdesaida  MCEP[1]
corpodocanal SENDERRECEIVER
interacaodeentrada  DATREQ
interacaodesaida  DATIND
[fim_vazao]
```

Para calcular o *delay* de uma determinada interação é necessário monitorá-la desde o momento em que esta é enviada até o momento em que ela chega ao seu destino, para isso o compilador deve gerar as seguintes informações:

i) Corpo do Pai: Esta informação diz respeito ao módulo-pai dos módulos que estão se comunicando, respeitando as regras de escopo de Estelle;

ii) Cabeçalho: Esta informação junto com a informação do corpo do pai definem qual o módulo está sendo referenciado;

iii) Ip de Entrada: Define o ponto de interação no qual a interação está sendo depositada. O tempo começa a ser contado quando a interação é enviada através deste ponto;

iv) Ip de Saída: Define o ponto de interação no qual a interação será retirada. O tempo é calculado quando a interação chega a este ponto;

v) Corpo do Canal: Descreve o canal onde as primitivas foram declaradas;

vi) Interação de Entrada: Define a primitiva que está sendo usada para o envio da interação;

vii) Interação de Saída: Define a primitiva utilizada para a recepção da interação.

Já o cálculo da vazão requer, além dos ítens descritos como sendo necessários ao *delay* as seguintes informações:

i) Tempo de Medida: Esta informação define a quantidade de tempo de simulação que deve ser considerado no momento do cálculo da vazão;

ii) Tamanho dos Pacotes: Esta informação define em bytes o tamanho do pacote que está trafegando no meio de comunicação.

iii) Tempo de Rajada: Determina o período a ser considerado pelo simulador para efetuar o cálculo da vazão naquele período numa tentativa de monitorar rajadas de tráfego em um curto espaço de tempo.

Em alguns protocolos não é necessário a monitoração de uma interação em especial, pois o que se deseja é determinar quando um determinado evento ocorreu. A ocorrência de um determinado evento pode ser traduzida em disparos de transições nas especificações ESTELLE. Portanto, a partir do momento que detectamos quando uma determinada transição disparou, temos uma indicação da ocorrência de um determinado evento. Então, para monitorarmos o tempo da ocorrência de um evento, basta monitorarmos os tempos de disparos das transições envolvidas neste evento.

A FI Tempo-Real previu também um modelo que se adequa a medição de *delay* entre eventos. Para as informações relacionadas com os modelos que têm restrições de *Delay* entre eventos, a FI Tempo-Real segue o exemplo abaixo.

```
[delay_de_trans]
transdeinicio THOFF
transdefim THJOINCOMP
corpodeinicio MT_BODY
corpodefim MT_BODY
modvardeinicio MT
modvardefim MT
[fim_delay_de_trans]
FIM
```

É mostrada a seguir a descrição das informações associadas ao cálculo do *delay* entre eventos, ou *delay* de transições que requer as seguintes informações:

i) Transição de Início: Esta informação define a partir do disparo de qual transição o tempo começa a ser contabilizado;

ii) Transição de Fim: Esta informação define o evento, ou seja, o disparo de qual transição marca o fim da contagem de tempo;

iii) Corpo da Transição de Início: Define o corpo ao qual a transição de início pertence.

iv) Corpo da Transição de Fim: Define o corpo ao qual a transição de fim pertence.

v) Variável Módulo da Transição de Início: Define a variável módulo ao qual a transição de início pertence.

vi) Variável Módulo da Transição de Fim: Define a variável módulo ao qual a transição de fim pertence.

Com isso conclui-se a descrição da FI Tempo-Real que fornece todos os dados estáticos relacionados às restrições temporais.

3.5 Mecanismo utilizado para monitorar PDU

Recapitulando a especificação de uma restrição temporal utilizada na seção 2.5, com base na figura 3.3, uma restrição de *delay* seria:

```
   TIME CONSTRAINTS

NAME delay-maximo:
FORALL t : time;
FORALL z : integer;

  HENCEFORTH(
   SENDING_OF MCEP[0].DATReq AT t LEADSTO
    RECEIVING_OF MCEP[1].CONReq AT ( now <= t + 20)
```

Podemos observar que as restrições temporais são feitas com base em interações, ou primitivas de serviço. Para o simulador, isso funciona bem e é suficiente desde que o meio seja modelado sem perdas, daí então pode-se garantir que a a primeira primitiva CONReq que chegue ao receptor pelo ponto MCEP[1] é realmente a primeira DATReq enviada pelo emissor no ponto MCEP[0] e assim sucessivamente, poderiamos então monitorar apenas as primitivas para calcular os tempos de cada mensagem. Porém, quando o meio de comunicação é modelado com perdas, a seqüência descrita não pode ser garantida.

Na figura 3.3 é mostrado o envio de uma primitiva, DATReq, em um ponto de interação e sua corresponte do outro lado tem o nome de CONReq. Quem realmente sai de um ponto de interação e trafega até o outro ponto é o parâmetro da primitiva, ou PDU (*Protocol Data Unit*). Portanto, quem deve ser monitorada fim-a-fim é a PDU, e não a primitiva de serviço.

Para avaliar os requisitos quantitativos de tempo entre os módulos, foi criado um mecanismo que possibilitasse monitorar a trajetória de uma mensagem, ou PDU, fim-

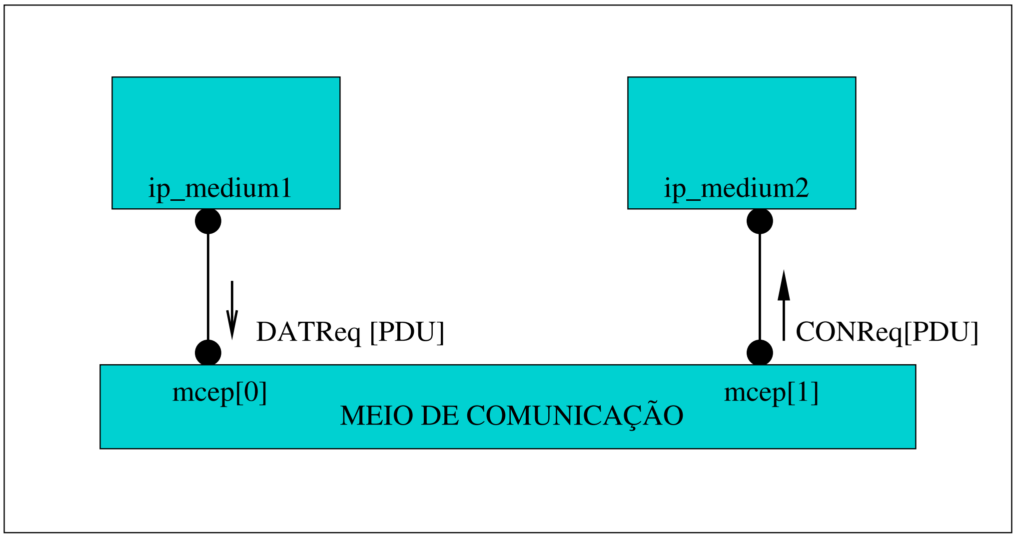

Figura 3.3: Primitivas de serviços diferentes transportam a mesma PDU

a-fim. Um cabeçalho é adicionado ao parâmetro contido em cada interação que contenha restrição temporal. O cabeçalho contém apenas um campo com o número de seqüência da mensagem. Veja a figura 3.4

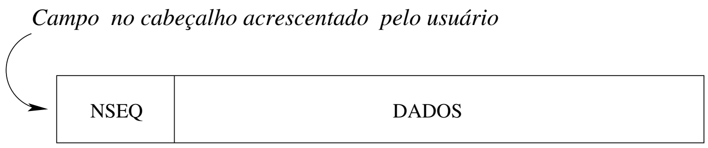

Figura 3.4: Cabeçalho acrescentado à PDU

A criação do cabeçalho é função obrigatória do usuário no momento da especificação Eetelle, por meio da declaração de um campo no parâmetro da interação. O número de seqüência deve ser declarado com o identificador *nseq*. No sentido de exemplificar uma forma de declaração, logo a seguir, pode-se ver uma PDU especificada com o uso de uma estrutura de dados do tipo RECORD.

```
pdu_type = RECORD
          C    : codigo_pdu_type;
          info : sdu_type;
```

```
          nseq : INTEGER;
           {declaração do número de seqüência}
         END;
{colocar na interface variaveis}
```

Nesse exemplo, *nseq* foi incluído nos campos da PDU , após os campos usados de fato pelo protocolo especificado. O campo *nseq* é utilizado (consultado e alterado) pelo simulador. O usuário deve ainda, colocar essas declarações na *InterfaceVariaveis* que é um procedimento de uso do simulador utilizado para consultar e alterar campos que são parâmetros da PDU, que neste caso é o próprio campo *nseq*, veja o exemplo a seguir:

```
          .
          .
          .

body SenderBody for SenderType;
        var
          m:pdu_type;

           (* insercao dos parametros da interacao
         no InterfaceVariaveis *)


   {$afetar channel Medium_interface DATREQ := m.nseq}
           .
           .
           .

    body ReceiverBody for ReceiverType;

        var
          m:pdu_type;

           (* insercao dos parametros da interacao no
          InterfaceVariaveis *)
```

```
{$verificar channel Medium_interface DATIND m.nseq}

                .
                .
                .
```

A diretiva ($afetar) é utilizada para alterar valores de parâmetros da PDU, isto é feito no envio da PDU e a diretiva ($verificar) é utilizada apenas para consultar o valor do parâmetro declarado.

A única alteração feita pelo simulador no parâmetro da interação é preencher o campo *nseq* com o valor atual do número de seqüência, isto é feito no momento do envio da interação. Depois disso, o simulador apenas consulta o campo adicionado no momento da recepção.

Sempre que uma interação que sofre alguma restrição temporal for enviada, neste momento o simulador cria um estrutura de dados para armazenar os dados referentes àquela PDU, como mostra a figura 3.5

No momento do envio apenas dois campos da estrutura de dados são preenchidos, o *nseq* e o tempo de simulação em que aquela PDU está sendo enviada. No momento da recepção de uma PDU, é verificado qual o valor do campo *nseq* que faz parte do parâmetro da PDU e então faz-se uma varredura na estrutura de dados para que se possa preencher os outros campos, como o *delay*, por exemplo.

3.5.1 Variáveis Temporais

A seguir, uma descrição das variáveis e fórmulas que regem o cálculo das condições temporais: T_d, T_e, T_{now}, T_t.

T_d é o tempo de disparo de uma transição. Este valor é escolhido de maneira aleatória, dentro de uma faixa de valores definidos na especificação pelo projetista, segundo a semântica do Sistema de Transições Temporizadas[11]. No caso de haver várias transições espontâneas sensibilizadas, que fazem uso da cláusula *Delay*, foi criado um intervalo de disparo baseado na semântica de Redes de Petri Temporizadas [24], que será utilizado para fazer a escolha aleatória do tempo dentro desse intervalo, T_d. Por exemplo, considere três transições sensibilizadas com os seguintes valores de *Delay*:

```
T1 = ( 3, 8 )
```

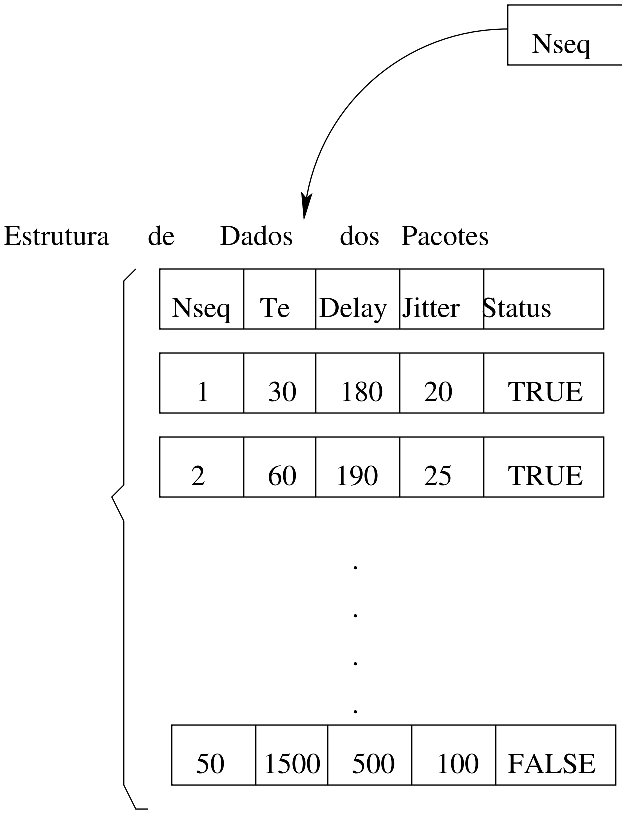

Figura 3.5: Estrutura de dados criada pelo simulador

```
T2 = ( 2, 9 )
T3 = ( 2, 7 )
```

O intervalo de disparo é formado pelo mínimo dos limites inferiores e o mínimo dos limites superiores. Portanto, o intervalo de disparo do exemplo acima seria: ID = (2, 7). Só então, a transição a ser disparada será escolhida, também de maneira aleatória.

O valor de tempo selecionado, supondo o valor de T_d= 3, será decrementado dos limites inferiores e superiores das outras transições que não foram disparadas. Após o disparo de uma transição é refeito o cálculo de quais novas transições estão sensibilizadas. Se as transições T1, T2 e T3 ainda permanecerem sensibilizadas, o intervalo delas será

```
T1 = ( 0, 5 )
```

```
T2 = ( 2, 9 )
T3 = ( 0, 4 )
```

Observe que a transição T2 que foi disparada, permanece com o *delay* original, já que ela foi disparada e voltou a ficar sensibilizada, ao contrário das outras que simplesmente mantiveram o estado de sensibilizadas.

A atribuição de intervalos de tempo de disparo de transições (*delay*) é feita para tentar retratar de maneira fiel o que acontece em um sistema real em que o tempo para a execução de uma determinada tarefa pode ser maior ou menor, dependendo da carga momentânea da CPU, ou seja, da quantidade e da complexidade dos processos que estão sendo executados. Depende ainda da capacidade de processamento do sistema alvo.

O usuário projetista é o responsável pela estimativa dos tempos de disparo das transições e, para tanto, o mesmo deverá possuir a percepção adequada para essa tarefa com base na sua experiência.

T_e é o tempo de simulação que um determinado pacote ou mensagem foi enviada.

T_{now} é o tempo de simulação atual.

No momento do envio de uma interação, o campo T_e é feito igual a T_{now}. $T_e = T_{now}$;

À medida que a simulação prossegue, o valor T_{now} é incrementado pelos tempos de disparo de outras transições, T_d.

3.5.2 Cálculo do *Delay*

O cálculo do tempo de simulação, T_{now}, obedece a seguinte fórmula:

$$T_{now} = \sum_{1}^{n} Td_n; \tag{3.1}$$

Ou seja, é o somatório dos tempos de disparo de cada transição. Os tempos entre o disparo de uma e outra transição, bem como o tempo de disparo de transições que não fazem uso da cláusula *delay* de Estelle são desprezíveis e, portanto, desconsiderados do cálculo.

Na figura 3.6 temos uma simulação de um cálculo de atraso. Utilizamos apenas um estado e uma transição para representar a máquina de estado de cada camada, com uma faixa de tempo associada à cada transição. No momento do envio da interação o eixo do tempo indicava T_{now} igual a 10, neste momento o valor de T_{now} é armazenado em Te (tempo de envio da interação). Durante o andamento da simulação os tempos de disparo

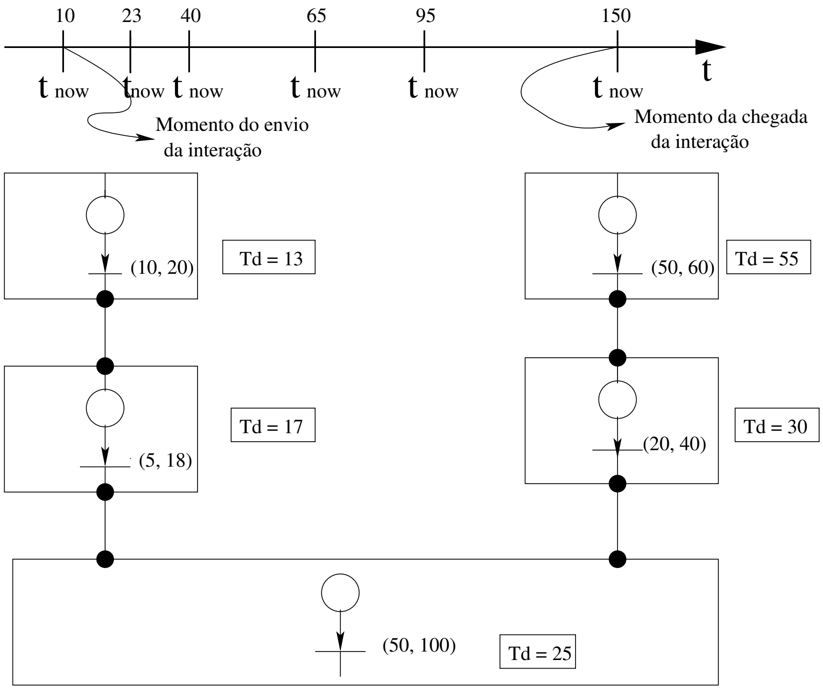

Figura 3.6: Simulação de Cálculo de um Atraso

de cada transição, representados por Td, e que são escolhidos de maneira aleatória dentro do intervalo especificado vão incrementando o valor de T_{now}.

Quando o pacote chega ao seu destino, a diferença entre T_{now} e T_e, revela o atraso total de percurso, T_t, daquele pacote, como segue:

$$T_t = T_{now} - T_e; \tag{3.2}$$

No nosso exemplo, no momento em que a interação chega ao seu destino, T_{now} marca 150 unidades de tempo, usando a equação 3.2, temos que o atraso daquela interação é 140 unidades de tempo.

De posse dos números de seqüência de cada pacote e dos tempo de envio e de chegada, o simulador monta uma tabela com tais informações e a fornece ao usuário como mais uma fonte de dados.

3.5.3 Cálculo do *Delay* entre Eventos

O *delay* entre eventos, que é caracterizado pelo disparo de duas transições, é uma característica importante fornecida pelo simulador, já que abre um grande leque de possibilidades na especificação de vários tipos de mecanismos em protocolos, como exemplo, pode-se citar, fases de estabelecimento e encerramento de conexões. Além das variáveis já descritas na seção 3.5.1 mais duas variáveis temporais foram criadas para tratar o atraso entre os eventos, T_{start} e T_{stop}, elas guardam os tempos de início e fim da contagem de tempo entre os eventos, respectivamente. No momento do disparo da primeira transição selecionada, o valor do T_{now} é guardado em T_{start} e ao final, marcado pelo disparo da segunda transição selecionada, T_{now} é armazenaado em T_{stop}, a diferença entre T_{stop} e T_{start} revela o atraso total entre os eventos. É importante destacar que para monitorar este tipo de evento não é necessário incorporar o campo *nseq* à PDU como descrito na seção 3.5, pois ele monitora apenas os disparos das transições.

3.5.4 Cálculo do *Jitter*

O *Jitter* é a variação do atraso com o qual os pacotes chegam ao receptor. O cálculo do *Jitter* é sempre feito considerando dois pacotes consecutivos. Sejam T_1 e T_2 os respectivos tempos que os pacotes gastaram, então o *jitter* dos pacotes 1 e 2 seria: $T_1 - T_2$.

Portanto, o cálculo do *jitter* é realizado após o término do cálculo do *delay* de cada pacote. Para efeito de gráfico, assumimos que o *jitter* do primeiro pacote é zero e o *jitter* do segundo é sempre a diferença do atraso em relação ao primeiro, do terceiro em relação ao segundo e assim por diante.

3.5.5 Cálculo da Vazão

A vazão mede efetivamente a taxa de transmissão líqüida, ou seja, é a diferença entre a quantidade de pacotes que está sendo enviada pelo transmissor e a quantidade que está sendo perdida no meio de transmissão.

O cálculo da vazão média é feito considerando a quantidade de bytes que chega ao receptor divido pelo tempo de medida definido na FI Tempo-Real. Caso o simulador encontre o ponto de parada antes que o tempo de medida seja atingido então o cálculo é efetuado considerando o tempo total de simulação até o momento, que é dado pelo T_{now}. Portanto, assumindo que o tempo está em milissegundos, a vazão média é dada por:

$$V = \frac{Qbytes * 1000 * 8}{tempo\_de\_medida} \tag{3.3}$$

Sendo Qbytes calculado através do produto do tamanho de cada pacote pelo número de pacotes que chegaram ao receptor. O tamanho dos pacotes deve ter um valor fixo, tendo sido adotado 1500bytes como valor *default* por ser o valor mais utilizado para *MTU* de redes *Ethernet*, portanto $Qbytes = n * 1500$, onde n é o numero de pacotes recebidos. Esta limitação se deve ao fato de que em Estelle Tempo-Real não há como especificar tamanhos diferentes para PDUs iguais.

A vazão instantânea também é calculada a fim de monitorar possíveis rajadas que é a característica principal das mídias contínuas. Neste caso o usuário deve fornecer um intervalo de tempo que será monitorada a rajada. O gráfico da vazão também é fornecido pelo simulador.

3.5.6 Cálculo da Taxa de Perdas

A taxa de perdas mede o percentual de pacotes que está sendo perdido no meio de comunicação. É calculada a partir da subtração entre a quantidade de pacotes transmitidos e a quantidade de pacotes recebidos dividido pelo total de pacotes transmitidos, como segue:

$$Taxa\_de\_Perdas = 100.0 * \frac{pacotes\_transmitidos - pacotes\_recebidos}{pacotes\_transmitidos} \% \tag{3.4}$$

3.5.7 Avaliação da Restrição Temporal

O processo de avaliação do cumprimento de uma determinada restrição temporal consiste na chamada de procedimentos que serão gerados pelo compilador, tais procedimentos pedem os tempo de envio da mensagem e do tempo atual de simulação como parâmetros. Os procedimentos devem retornar o *status* daquela mensagem, ou seja, se a restrição foi atendida ou não. No caso do não cumprimento de uma determinada restrição, o usuário pode fazer uso da cláusula *Otherwise* para tomar uma nova decisão, tal cláusula permite ao usuário atribuir um novo comportamento ao protocolo. Ao simulador cabe apenas o papel de fornecer estas informações ao projetista para que este refaça ou altere o seu projeto. O simulador está preparado para chamar os procedimentos do compilador e a estrutura de dados para armazenar a resposta também está criada.

3.6 A Interface do Simulador

A interface foi construída visando facilitar o uso do simulador, sendo bastante intuitiva e tentando tornar mais fácil a interpretação dos resultados. São mostrados traços da simulação, permitindo uma verificação do andamento da simulação e gráficos ilustrando o comportamento dos parâmetros especificados. Fornecendo assim, uma maneira de se ter uma medida de desempenho do sistema.

Outro objetivo, foi tornar o acesso ao simulador possível através da Internet, permitindo que qualquer usuário em qualquer parte do mundo, pudesse fazer uso do sistema. Por conta disso, foi escolhida a linguagem JAVA, que permite que o usuário, utilizando *browsers*, possa carregar por *download* um programa java a partir de uma página da WEB e executá-lo localmente no seu próprio sistema. Estes programas são denominados *applets* [25]. Além disso, a portabilidade de JAVA faz com que o sistema não fique preso a nenhuma plataforma. A Interface é composta basicamente por botões que acionam a execução de uma determinada tarefa e por uma área de texto onde serão mostrados os traços da simulação e o seu andamento, indicando quais transições estão sendo disparadas bem como o tempo de simulação em que elas estão sendo disparadas, além de outras informações solicitadas pelo usuário.

O usuário faz a seleção das opções através de um pequeno campo de texto para em seguida pressionar o botão *OPÇÃO*. Outra informação que deve ser fornecida pelo usuário é o numero de passos que o simulador irá executar, o que equivale a quantidade máxima de transições que serão disparadas, pois pode acontecer da simulação atinjir o seu final antes de completar o número de passos solicitados pelo usuário. Para inicializar a simulação sem que haja qualquer *deadlock* é necessário que o usuário insira a primitiva adequada na fila associada ao ponto de interação correspondente. Isto pode ser feito selecionando a opção [1] na interface, veja o trecho abaixo:

```
*** Alteracoes de Fila ***
[1] Inserir Interacao
[2] Retirar Interacao
[3] Retirar Todas as Interacoes
[4] Modificar Parametro
Opcao(0 para desistir) :

****** Opcoes de Instancia de Modulo *******
[1] ==>SYSSNDRECV[0]
[2] ==>SENDER[0]
[3] ==>RECEIVER[0]
```

```
[4] ==>MEDIUM[0]
Opcao(ou 0 para desistir) :
[1] ==>UCEP[0]
[2] ==>MCEP[0]
Opcao(0 para desistir) :

****** Opcoes de Interacao *******
[1] ==> START
[2] ==> STOP
 *******************************************
 Voce desejar inicializar mais algum modulo ?
 *******************************************
 *******************************************
 Se deseja, click em SIMULAR
 *******************************************
 Se nao, click em CONTINUAR
 *******************************************
 *******************************************
 Escolha o numero de passos do simulador
 *******************************************
```

Este trecho mostra as seleções iniciais a serem feitas pelo usuário para dar início ao processo de simulação. Na seqüência:

i) Selecionar a opção [1] para inserir interação na fila;

ii) Selecionar o módulo responsável pela inicialização do sistema;

iii) Selecionar o ponto de interação no qual a interação será depositada;

iv) E por fim, selecione a interação que dará início à simulação.

Ao clicar no botão *READ-ME* pode ser encontrado um tutorial que faz um breve relato de como a ferramenta deve ser utilizada. Esse tutorial encontra-se disponível também no apêndice C. Veja a aparência da interface na figura 3.7

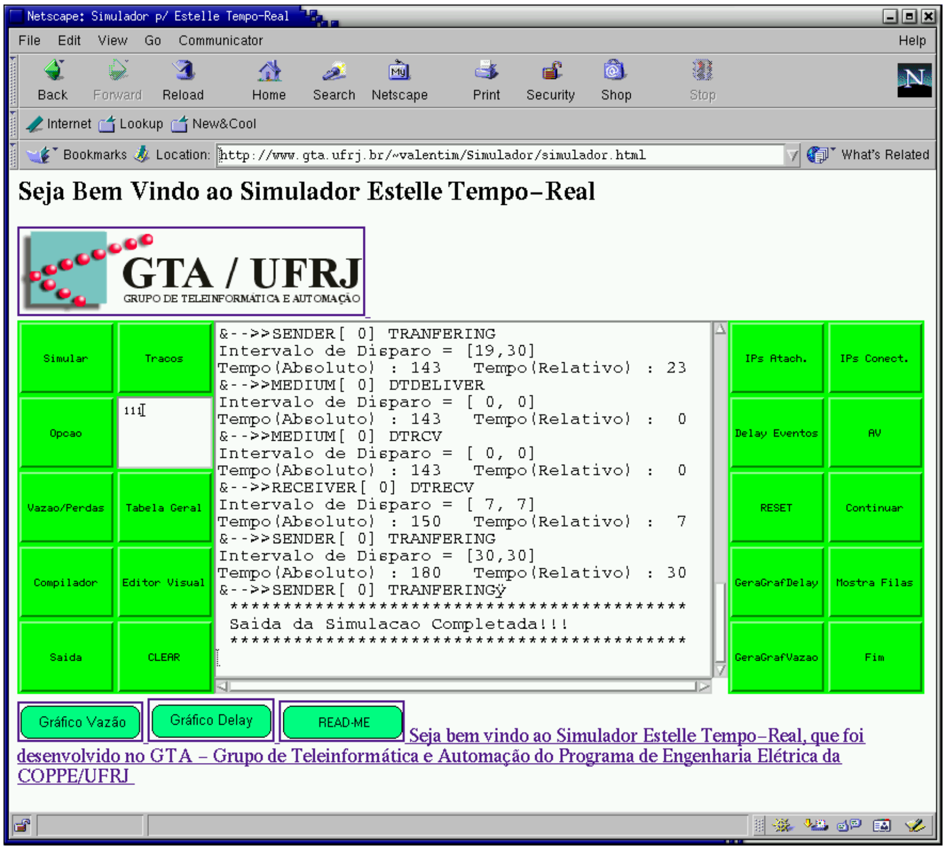

Figura 3.7: Interface Visual do Simulador

Como facilidades adicionais, a Interface permite:

- Obter informação sobre os pontos de interação *atachados* e conectados;
- *Resetar* dados de uma simulação corrente e dar início a uma nova sessão;
- Saída da simulação sempre disponível ao usuário;
- Conteúdo das filas associadas aos pontos de interação
- Gráficos de *Delay* e Vazão.

Os arquivos que armazenam os valores de calculados para *delay* e vazão também podem ser obtidos, permitindo que o usuário possa utilizar os pontos com uma outra ferramenta de geração de gráficos que atenda melhor às suas necessidades. O simulador utiliza a ferramenta *Gnuplot*[26] da distribuiçao *GNU/Linux*[1].

[1]http://www.ucc.ie/gnuplot/gnuplot.html

A figura 3.8 mostra um gráfico de *delay* gerado pela interface e mostrado em uma janela do *Browser*

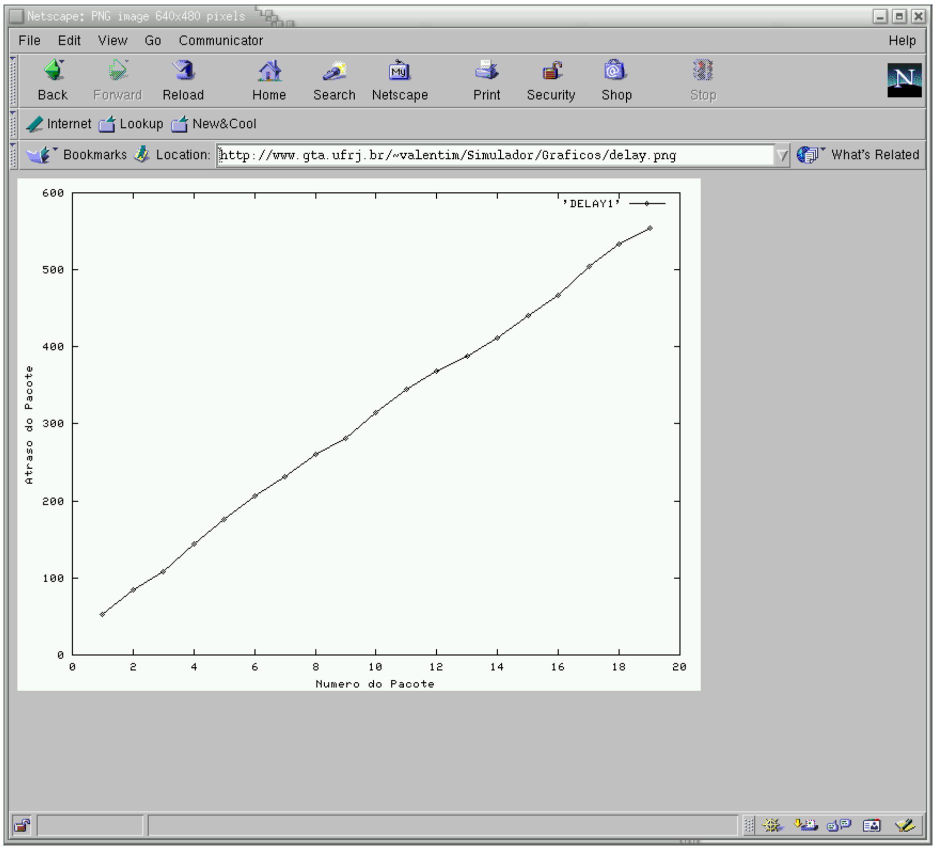

Figura 3.8: Gráfico gerado pela Interface

3.7 A Comunicação Cliente/Servidor

A comunicação entre a interface e o núcleo do simulador segue o paradigma cliente-servidor. Do lado cliente, a interface em código JAVA. Do lado servidor, o núcleo do simulador em Módula2. Isto foi possível porque o compilador Mocka [27], para Módula2, permite que módulos construídos em algumas outras linguagens, denominados *Foreign Module*, sejam executados a partir do código em Módula2. Este recurso permitiu que a comunicação entre a interface e o núcleo do simulador fosse feita através de *sockets* [28], escrito em JAVA, no lado cliente e escrito em C, no lado servidor.

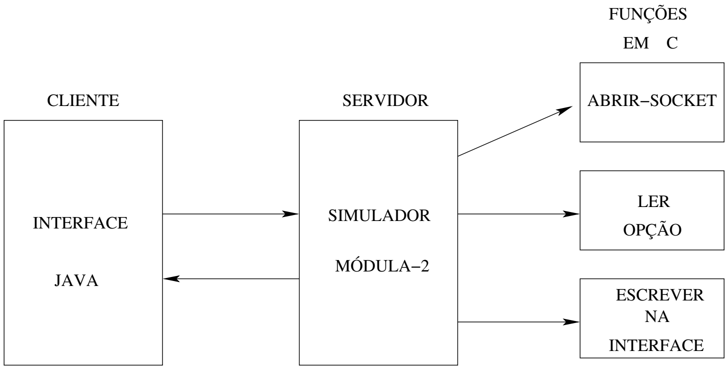

Figura 3.9: Comunicação entre a Interface e o Simulador

Esse modo de comunicação influenciou no desenvolvimento da interface, pois o desempenho do sistema como um todo depende principalmente dos níveis de congestionamento do enlace de comunicação. Então optou-se por diminuir ao máximo a quantidade de interações entre o cliente e o servidor, conseqüentemente aumentando a quantidade de informação a ser trocada em cada envio de dados de um ponto ao outro da conexão.

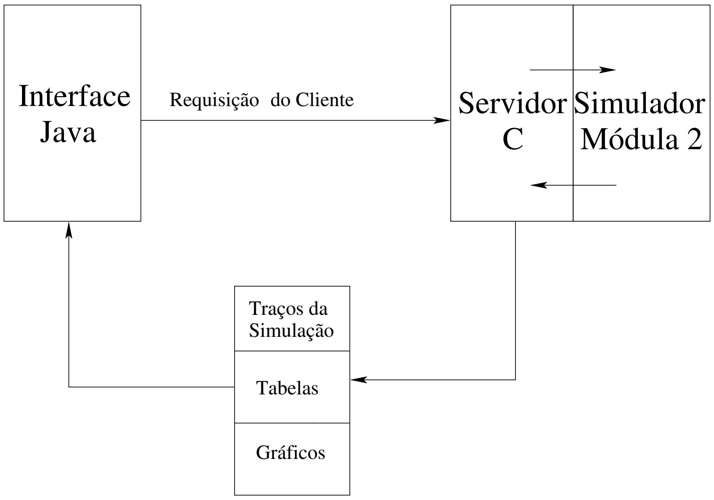

Figura 3.10: Comunicação entre a Interface e o Simulador

A figura 3.10 ilustra o modo como a interface do cliente e o simulador se relacionam. O cliente envia uma requisição solicitando uma tarefa, o servidor a executa e devolve a resposta ao cliente. Neste caso as respostas são traços da simulação, tabelas informativas e gráficos.

Estas informações são úteis para permitir ao usuário uma análise mais completa do desempenho dos parâmetros de tempo que garantem a qualidade de serviço da aplicação especificada, e se for o caso modificar o projeto.

3.8 O Ambiente de Implementação

O ambiente escolhido para a implementação do ProtCAD/RT foi o sistema operacional Linux, por se tratar de uma plataforma de distribuição gratuita e com código aberto, que possui versão para ser utilizada em computadores da linha Intel e compatíveis(a partir do 386). Isto minimiza os riscos de que sejam lançadas novas versões incompatíveis com as anteriores, tornando o sistema obsoleto.

A facilidade de encontrar suporte e documentação abundante na própria Internet e por ter se mostrado um sistema bastante eficiente e estável, também contribuiram para a sua escolha. Além disso, é notória a rápida difusão deste sistema operacional no meio acadêmico e comercial.

No desenvolvimento das interfaces, JAVA foi a linguagem eleita por garantir portabilidade, fator essencial para um sistema que pretende ser utilizado por pessoas em qualquer parte do mundo, usando hardwares diversos.

3.9 Conclusões

Este capítulo descreveu a ferramenta ProtCAD/RT de maneira geral, em particular o Simulador para Estelle Tempo-Real. Definiu a FI Tempo-Real que deve ser gerada pelo compilador, abordou o mecanismo utilizado para monitorar as mensagens fim-a-fim e como as restrições temporais são calculadas. A interface e a sua comunicação com o núcleo do simulador também foram descritas. No capítulo seguinte serão mostrados os resultados de alguns experimentos realizados, mostrando trechos dos traços de simulação obtidos e os gráficos de *delay* e vazão.

Capítulo 4

Experimentos e Resultados

4.1 Introdução

Neste capítulo serão apresentados alguns modelos de protocolos que foram especificados em Estelle com restrições temporais fim-a-fim, apresentando como resultados os traços da simulação, os valores calculados para vazão média, taxa de erros e os gráficos de *delay* e de vazão instantânea, sendo estes resultados obtidos a partir da interface visual acessada por um *browser*

O primeiro modelo, descrito na seção 4.2, será o clássico Produtor-Consumidor. O *buffer* compartilhado é infinito, portanto, não há perdas de pacotes. O transmissor (ou produtor) envia pacotes a uma taxa maior do que o receptor (ou consumidor) é capaz de tirar do meio de comunicação ou *buffer*.

As restrições impostas são o *delay* fim-a-fim dos pacotes e a vazão média no receptor.

A seção seguinte, 4.3, descreve um sistema CBR com perdas, onde um emissor a taxa constante gera pacotes para um receptor. Aqui será visto o gráfico de atraso por pacote e gráfico da vazão no receptor, além disso trechos dos traços de simulação obtidos também são fornecidos. A seção 4.4 descreve um arquitetura Multiponto-Ponto, onde vários transmissores enviam pacotes para um único receptor, neste exemplo, verifica-se uma maior taxa de perda e de atraso para os transmissores. Gráficos e trechos de saída da simulação são fornecidos. O último exemplo é um protocolo para controle de *Handoff*, descrito na seção 4.5, este protocolo permite verificar o cálculo de atraso entre eventos. A seção 4.6 faz alguns comentários finais.

4.2 O Modelo Produtor-Consumidor

O presente sistema apresenta uma arquitetura muito simples, onde tem-se um módulo emissor que gera pacotes a uma taxa variável, dependendo do valor de tempo sorteado para o disparo da transição cuja a ação é enviar pacotes ao emissor. Existe um módulo receptor responsável apenas pela recepção dos pacotes gerados pelo emissor. O meio de comunicação recebe os pacotes do emissor, e os repassa ao receptor com um atraso também aleatório. Desta forma, simulamos problemas que existem em sistemas reais, em virtude da limitação de banda dos enlaces de comunicação e filas nos *buffers* dos roteadores de uma rede.

As mensagens ou primitivas de serviço em Estelle Tempo-Real estão relacionadas às funções de transmissão e recepção de dados, bem como ao início e fim desta transmissão. Como primitivas do módulo emissor, tem-se:

- *START*: pedido de início de transmissão pelo usuário ao emissor;
- *STOP*: pedido de finalização de transmissão pelo usuário ao mesmo emissor.

Portanto, para dar início a simulação deste exemplo o usuário deve primeiro colocar a primitiva *START* no ponto de interação UCEP do módulo emissor para que o processo de simulação seja iniciado, esta tarefa é feita através da interface.

Para o meio de comunicação, existem apenas a primitiva de pedido de envio de dados pelo emissor (*DATREQ*) e a primitiva de concretização da entrega do pacote ao receptor (*DATIND*). O serviço prestado neste caso é sem conexão e sem confirmação. Como o emissor não presta serviço a nenhuma entidade em particular, ele não possui primitivas associadas ao módulo que expressa seu comportamento. Pode-se visualizar a arquitetura modular do sistema em questão na figura 4.1.

Nesta figura, estão representados os canais de comunicação entre os módulos e os seus respectivos pontos de interação. Pode-se verificar a existência de um módulo-pai (*SysSndRecv*) que engloba os módulos *SENDER*, *RECEIVER* e *MEDIUM*, pois desta forma torna-se possível expressar restrições temporais fim-a-fim neste sistema. Estas devem ser descritas necessariamente no escopo do corpo de módulo pai, para que se tenha acesso a objetos de todos os módulos envolvidos, como conteúdo das filas nos IPs (pontos de interação) de módulos diferentes que estão se comunicando.

Através do ponto de interação do usuário (*USR*), é feita a chamada das primitivas do módulo emissor pelo usuário do sistema. Este encontra-se conectado pelo canal Sys_interface ao IP UCEP do emissor. O emissor por sua vez envia suas mensagens de

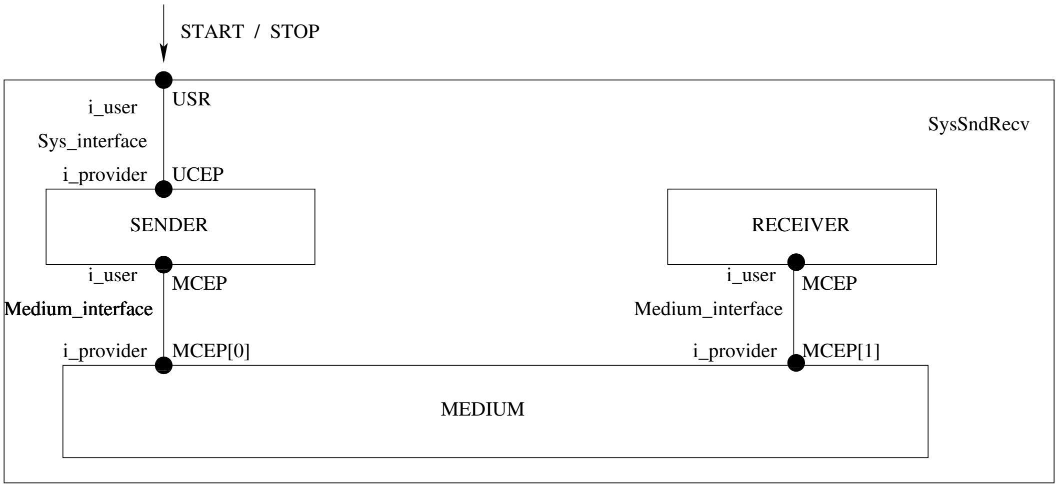

Figura 4.1: Modelo de Serviço

pedido ao meio pelo IP MCEP, conectado pelo canal Medium_interface ao IP MCEP[0] do meio. Da mesma forma, o meio entrega os pacotes de dados ao receptor, pelo IP MCEP[1], que se conecta ao IP MCEP do receptor por um outro canal Medium_interface.

4.2.1 Módulos Emissor e Receptor

A figura 4.2 mostra o modelo, em Máquina de Estados Finita (MEF), do comportamento do módulo emissor. Este emissor inicia sua execução em seu estado de repouso (*SenderIdle*). Quando ele recebe o pedido do usuário para iniciar a transmissão através da primitiva *START*, ele inicia a transmissão de seus dados, no estado *StartTransfer*, enviando em intervalos de aleatórios entre 0 e 30ms, através da mensagem *DATREQ*, onde o único parâmetro da PDU é o número de seqüência (*nseq*), que é acrescido a cada envio pelo simulador. O fato do transmissor enviar pacotes em intervalos aleatórios, faz com que a sua taxa de transmissão seja variável, portanto este emissor pode ser classificado como uma fonte VBR - *Variable Bit Rate*. O ciclo do transmissor é interrompido se o usuário entrar com um pedido de interrupção da transmissão (*STOP*) ou se o número de seqüência chegou ao máximo que existe para transmitir (*MAXSEQ*) ou então quando for atingido o número máximo de passos estabelecido pelo simulador. Após isto, o emissor volta ao repouso.

Como já foi mencionado, o receptor deste modelo é extremamente simples e possui um comportamento passivo. Desta forma, sua modelagem não necessita de uma MEF para sua representação, mas apenas de uma transição associada à recepção da mensagem

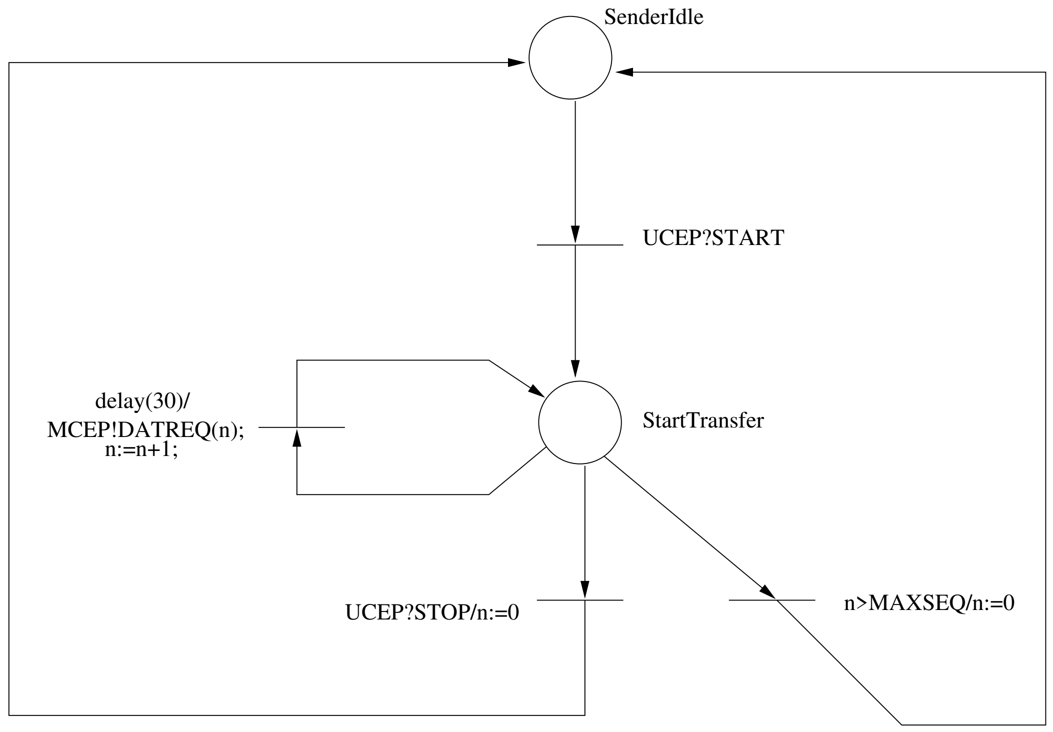

Figura 4.2: Comportamento do Módulo Emissor.

DATIND, que simula a chegada dos pacotes enviados.

4.2.2 Modelagem do Meio de Transmissão

Para especificar o meio com o comportamento desejado, foi criado um *buffer*, no qual são depositados os pacotes vindos da fonte, e o meio por sua vez, tem a função de retirar um a um esses pacotes do *buffer*, dar um atraso aleatório entre 50 e 200ms e enviá-los para o receptor.

O meio só retira um outro pacote do *buffer* depois de ter enviado o primeiro ao receptor, e este primeiro pacote é retido no meio por um tempo aleatório, dentro do intervalo já citado. Portanto, a taxa com que o meio retira os pacotes do *buffer* também é aleatória. Dessa forma, os pacotes terão valores de atraso diferentes quando chegarem ao receptor. Isto é feito para tentar retratar de maneira fiel o comportamento de um meio de comunicação real, no qual, o atraso na chegada de cada pacote ao receptor depende da situação momentânea de congestionamento da rede de comunicação.

A figura 4.3 retrata este comportamento. Ao receber o primeiro pacote do emissor para ser transmitido, ele é enfileirado, e sempre que chegarem novos pacotes estes também o serão.

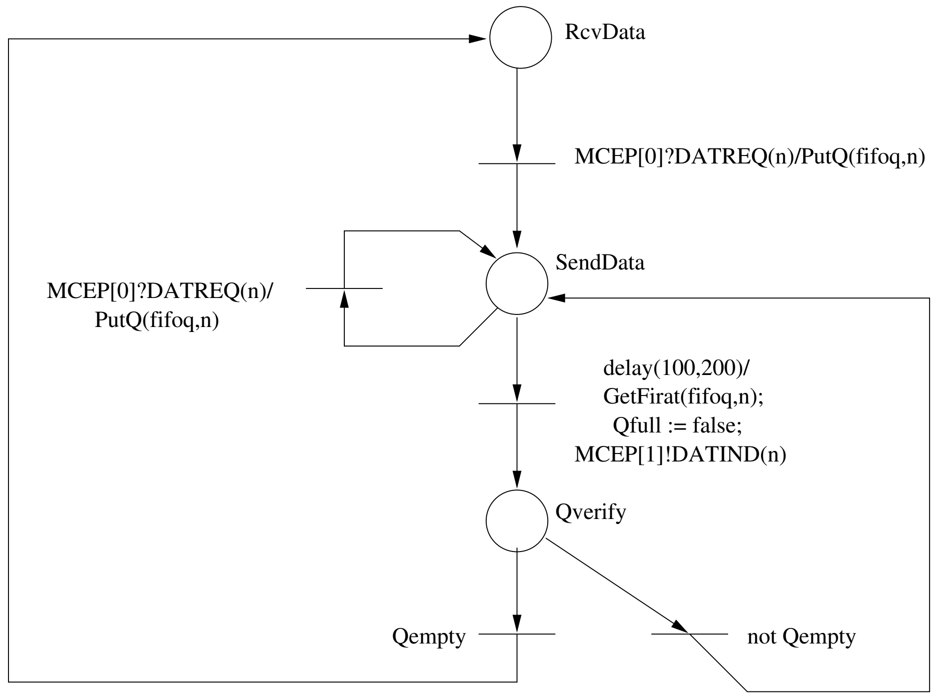

Figura 4.3: Comportamento do Meio de Comunicação.

4.2.3 Restrições Temporais do Sistema

Nesta seção, serão descritas as restrições temporais do sistema fim-a-fim, ou seja, essas restrições não são associadas a um módulo em particular, e sim, associadas à comunicação entre os diversos módulos que compõem o sistema. O trecho de código abaixo mostra a especificação do *delay* global do sistema. A combinação dos operadores *henceforth* e *leadsto* permite-nos expressar que, se um pacote é enviado em um tempo *t*, este pacote chegará ao receptor em um tempo $(now < t + MAX)$.

```
time constraints

    name delay :

        forall t:time;

        forall seqnr: integer;
```

henceforth (

sending_of SENDER.MCEP[0].DATREQ[seqnr] at t

leadsto receiving_of RECEIVER.MCEP[1].DATIND[seqnr]

at (now<t+MAX));

end;

end;

Desta restrição imposta, o compilador deverá gerar o seguinte trecho encontrado na Forma Intermediária Tempo Real:

```
[delay]
[d]
corpodopai SENDRECVBODY
cabecalho  MEDIUMTYPE
ipdeentrada MCEP[0]
ipdesaida MCEP[1]
corpodocanal SENDERRECEIVER
interacaodeentrada DATREQ
interacaodesaida DATIND
[fd]
[fim_delay]
```

Onde são descritos o nome do corpo pai dos módulos que estão se comunicando - SENDRECVBODY, respeitando as regras de escopo de Estelle, o cabeçalho do corpo ao qual pertencem os pontos de interação envolvidos na restrição - MEDIUMTYPE, os pontos de interação - MCEP[0] e MCEP[1], o corpo do canal no qual foram declaradas as primitivas utiizadas na comunicação - SENDERRECEIVER e por fim as próprias primitivas - DATREQ e DATIND.

Para o cálculo da vazão média além dos parâmetros já descritos para a restição *delay*, o tempo de medida (em unidades de tempo) e o tamanho do pacote (em octetos) que devem ser considerados no momento do cálculo são pré-definidos pelo compilador.

O campo "tempodejada" é utilizado para calcular a vazão instantâneamente.

```
[vazao]
```

```
tempodemedida 10000
tamanhodopacote 2000
tempoderajada 10
paidomeio  SENDRECVBODY
cabecalhodomeio MEDIUMTYPE
ipdeentrada MCEP[0]
ipdesaida  MCEP[1]
corpodocanal SENDERRECEIVER
interacaodeentrada DATREQ
interacaodesaida DATIND
[fim_vazao]
```

4.2.4 Resultados Obtidos

A figura 4.4 mostra um trecho dos traços de simulação obtidos mostrando os módulos envolvidos, as transições que foram disparadas, os pontos de interação e as interações enviadas, bem como o tempo de simulação em que o evento ocorreu. A listagem completa dos traços de simulação do produtor-consumidor pode ser encontrada no apêndice A.

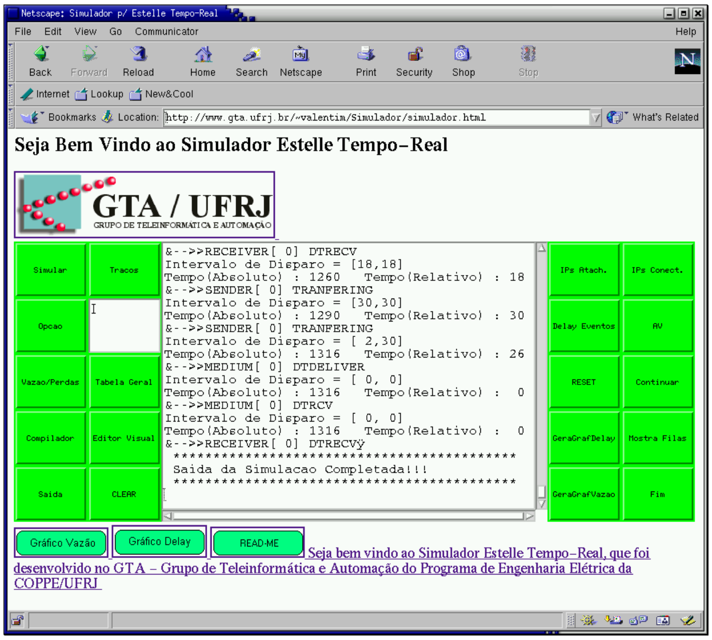

Figura 4.4: Traços de simulação mostrados na Interface.

O grafico na figura 4.5 mostra os valores de atraso por pacote para o exemplo do produtor-consumidor, observa-se que chegou a ser enviado até o pacote com número de seqüência 19, como já foi dito esse valor foi limitado pelo número de passos do simulador. Os valores de *delay* aumentam quase que de forma linear, visto que a medida que o emissor vai enviando pacotes e o receptor não os tira do meio de comunicação com a mesma velocidade, os últimos pacotes sofrem um atraso maior do que os primeiros, retratando assim uma situação de congestionamento como acontece nos sistemas reais, embora neste exemplo não tenhamos implementado perdas, o que significaria que o pacote foi descartado por o *buffer* do roteador já ter atingido a sua capacidade máxima de armazenamento. A perda será implementada no exemplo a seguir.

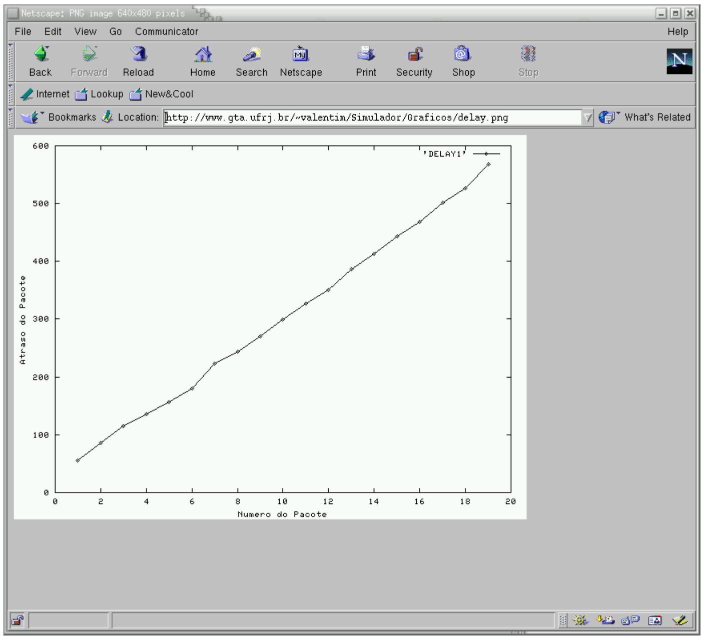

Figura 4.5: Gráfico de *Delay* por Pacote.

Outra informação mostrada ao usuário é o gráfico da vazão, plotando seus valores por curtos intervalos de tempo, que será definido pelo usuário, como tentaviva de visualizar as rajadas características de tráfegos como voz e vídeo. Veja a figura 4.6.

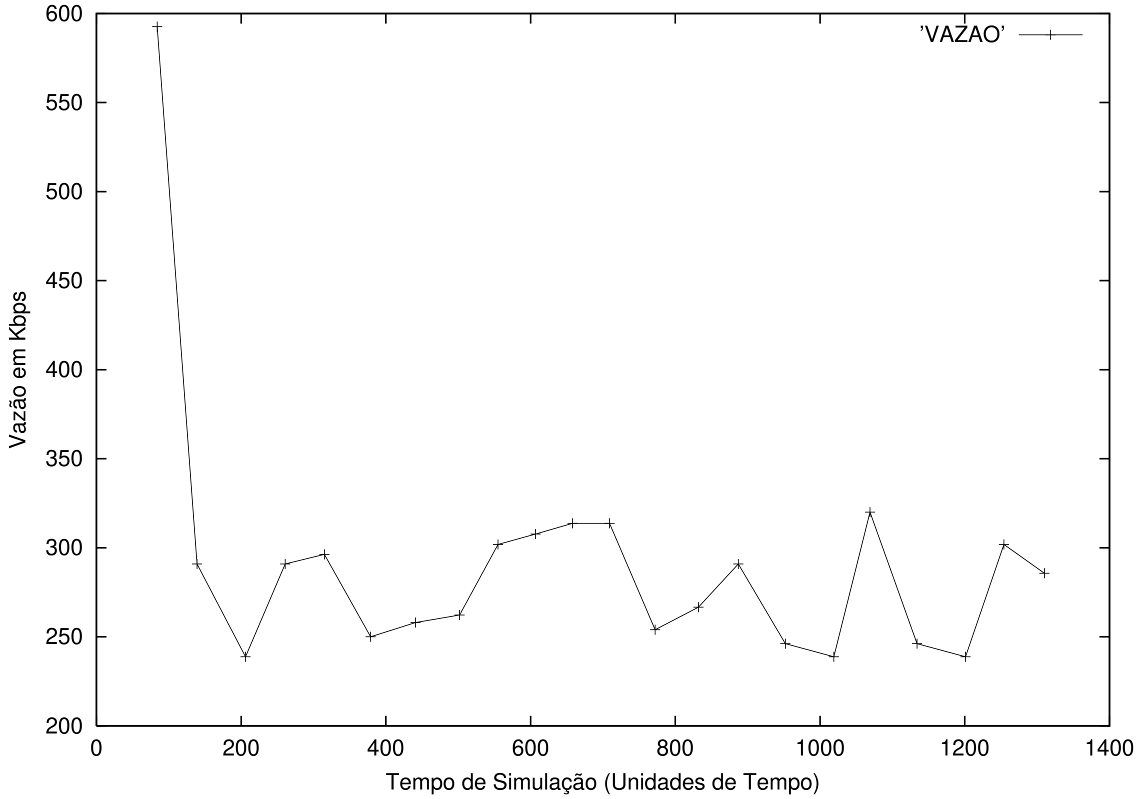

Figura 4.6: Gráfico da vazão por tempo de simulação.

4.3 Fonte CBR

Agora será apresentado um modelo semelhante ao exemplo anterior, porém o transmissor tem uma taxa de transmissão contante, o que o caracteriza como uma fonte CBR - *Constant Bit Rate* e o meio será modelado com perda de pacotes para tentar retratar de maneira fiel o que acontece nos roteadores. Quando o *buffer* enche, os próximos pacotes que chegam são descartados, os protocolos da camada de transporte que trabalham com controle de fluxo reduzem a taxa de transmissão quando percebem que o meio está congestionado tentando aliviar a carga dos roteadores como é o caso do TCP - *Transmission Control Protocol* até que se consiga dar vazão aos pacotes armazenados, porém, protocolos para mídias contínuas que são o nosso alvo de estudo não utilizam este tipo de controle, pois uma redução na taxa de transmissão afetaria gravemente o cumprimento da restrição *delay*, da restrição *jitter* e da própria vazão. O UDP *User Datagram Protocol* é um exemplo de protocolo sem controle de fluxo. A figura 4.7 mostra a arquitetura do Sistema CBR

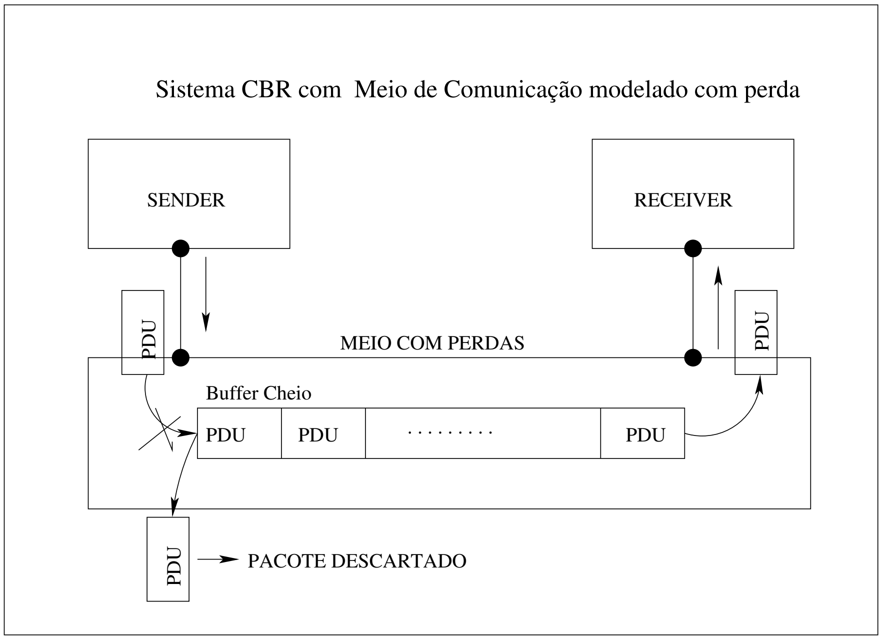

Figura 4.7: Sistema CBR com Perdas.

A forma intermediária que deve ser gerada pelo compilador é semelhante a mostrada na seção anterior. A figura 4.8 mostra os valores de atrasos por pacote, observe que a partir do pacote de número 20, começou a haver perdas, isto porque o *buffer* foi implementado com um tamanho de 15 pacotes e os valores de *delay* aumentam quase que linearmente no início até atingir o valor próximo do máximo que seria a soma dos atrasos de pelo menos 14 pacotes, isto acontece quando começam as perdas, a partir daí os pacotes que não são descartados encontram a fila praticamente cheia, por isso a proximidade dos valores de atrasos. Vale lembrar que o tempo gasto no processamento de cada pacote é o mesmo do exemplo anterior, entre 50 e 200 unidades de tempo

Por se tratar de uma fonte operando a taxa constante, era de se esperar que o gráfico da vazão se aproximasse de uma reta paralela ao eixo das abcissas, como pode comprovar a figura 4.9

4.4 Sistema Multiponto-Ponto

O terceiro exemplo utilizado foi sistema Multiponto-Ponto com três fontes enviando pacotes para um único emissor. O meio foi implementado com perdas. A arquitetura pode ser vista na figura 4.10

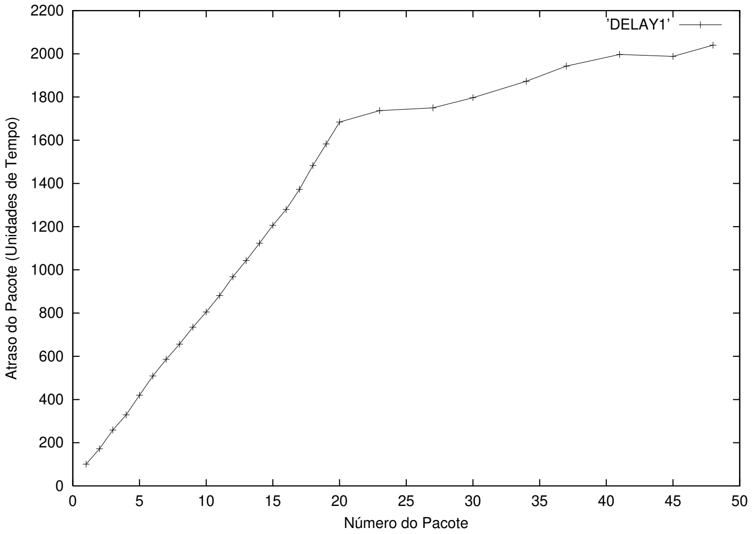

Figura 4.8: Sistema CBR com perdas - Atraso por pacote.

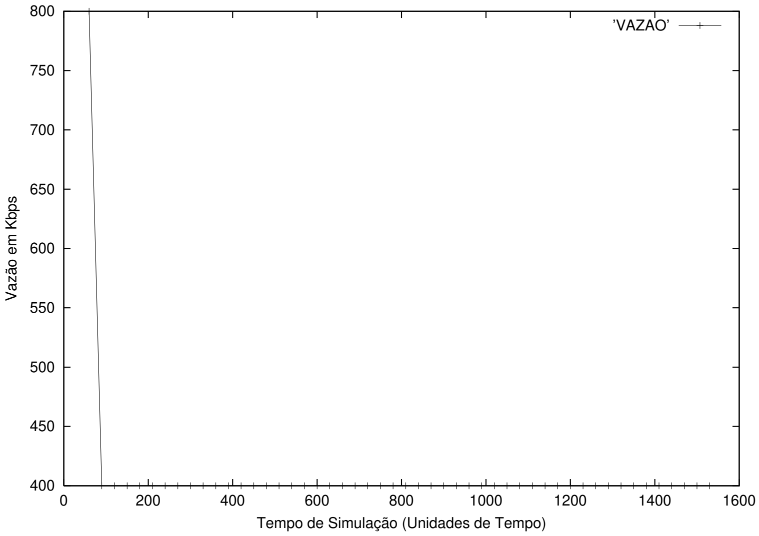

Figura 4.9: Sistema CBR com Perdas - Vazão instantânea.

Nesta arquitetura, o gráfico de atrasos por número de seqüência para cada emissor não deve sofrer grandes alterações, visto que as fontes são exatamente iguais. Portanto há a tendência dos pacotes encontrarem as filas praticamente cheias durante todo o tempo de simulação. Essa situação pode ser constatada nos gráficos a seguir.

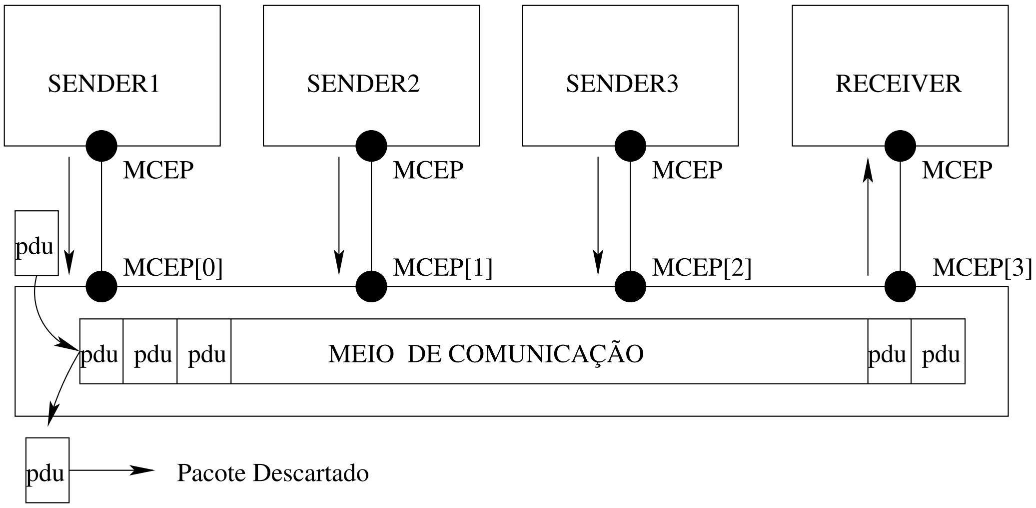

Figura 4.10: Arquitetura do sistema Multiponto-Ponto.

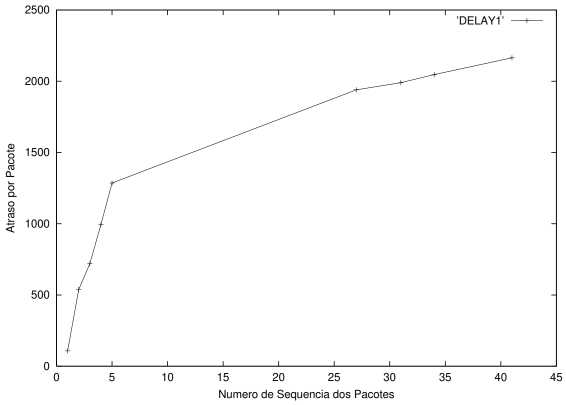

Figura 4.11: Atraso para o 1^o emissor

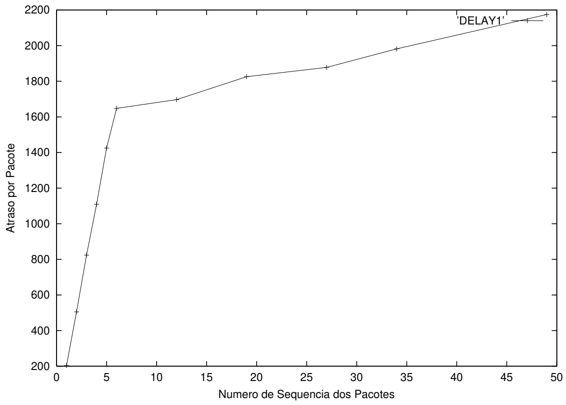

Figura 4.12: Atraso para o 2^o emissor

Porém, no que diz respeito a perda de pacotes, esta aumenta muito pois agora existe uma disputa pelo meio. As perdas podem ser visualizadas conferindo a ausência da marcação de pontos no gráfico. Uma alteração na cláusula *Delay* de qualquer um dos emissores tornando-o mais rápido do que os outros, se refletiria em um gráfico de atraso por pacote deste emissor com uma quantidade de perda menor, além disso seus pacotes teriam um atraso médio também menor.

O gráfico da vazão confirma a equivalência das fontes de transmissão, como mostra a figura 4.14

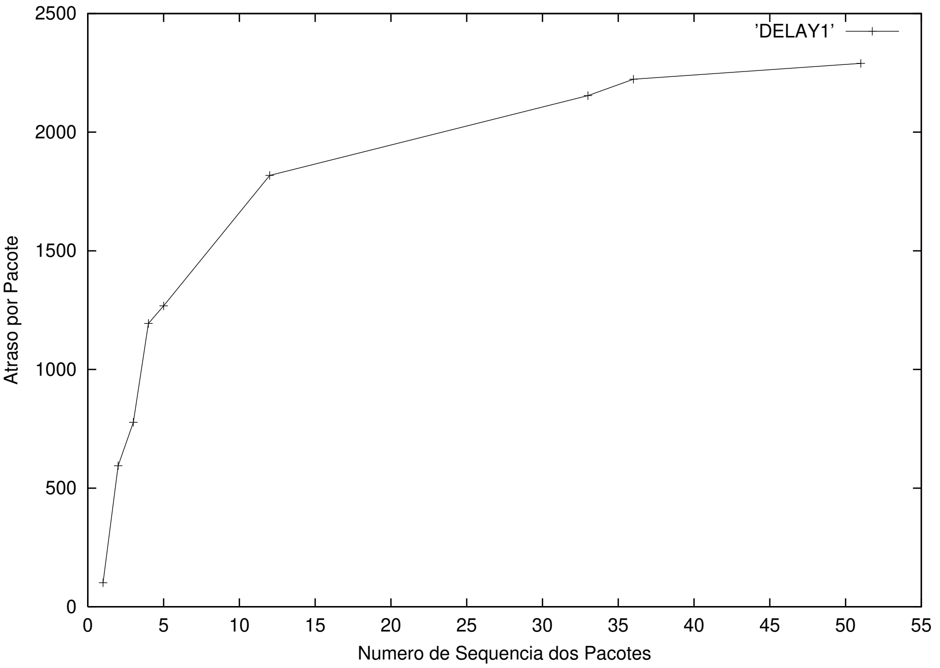

Figura 4.13: Atraso para o terceiro emissor

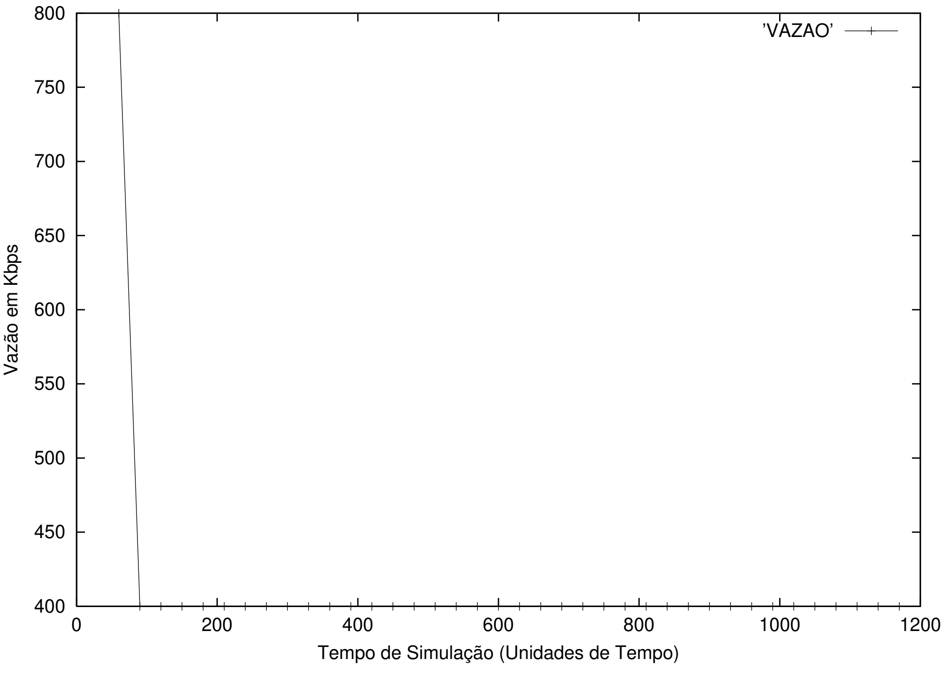

Figura 4.14: Vazão no receptor

4.5 Protocolo de Controle de Handoff

Nas redes tradicionais, quando uma conexão está estabelecida entre dois pontos de comunicação, assume-se que aquela topologia não sofre qualquer mudança durante o período de tempo de vida da conexão (exceto devido a falhas no *link* de comunicação).

Em redes móveis, o usuário necessita freqüentemente ter sua conexão refeita, visto que a todo momento ele está trocando de uma célula para outra. Este processo é conhecido como *Handoff*

A restrição se faz importante pois o processo de *handoff* deve acontecer dentro de um limite de tempo, principalmente se o sistema utilizado é o TDMA, cuja a limitação fica em torno de 500ms [29], o que significa que a supressão de uma conversação, por exemplo, pode durar no máximo 500ms, caso contrário poderia haver uma falha na comunicação. No sistema CDMA o processo é suave, conhecido como *soft-handoff*, aqui o usuário só é desconectado da primeira estação quando já estiver conectado à segunda.

O protocolo escolhido para especificarmos restrições temporais é uma proposta para ser usada em redes ATM sem fio [30].No modelo um terminal móvel (MT) sai da célula de responsabilidade da estação base1 (BS1) e passa para a estação base 2 - (BS2).

A arquitetura para o protocolo de controle de *Handoff* é mostrada na figura 4.15

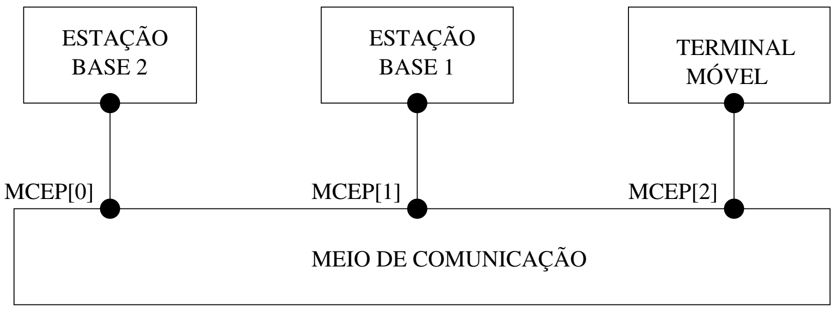

Figura 4.15: Arquitetura da Especificação Estelle Tempo-Real do Protocolo de *Handoff*.

O início do processo acontece quando o terminal móvel percebe queda no nível do sinal e envia a primitiva HANDOFF REQ para a estação BS1 que imediatamente envia a mesma primitiva só que agora contendo a PDU Setup, portanto, HANDOFF REQ(Setup) para a estação BS2. O processo termina quando a estação BS2 envia para o terminal móvel a primitiva HANDOFF JOIN. A máquina de estados finita completa descrevendo o comportamento das entidades está na figura 4.16

A tabela 4.1 descreve brevemente algumas das primitivas utilizadas no protocolo.

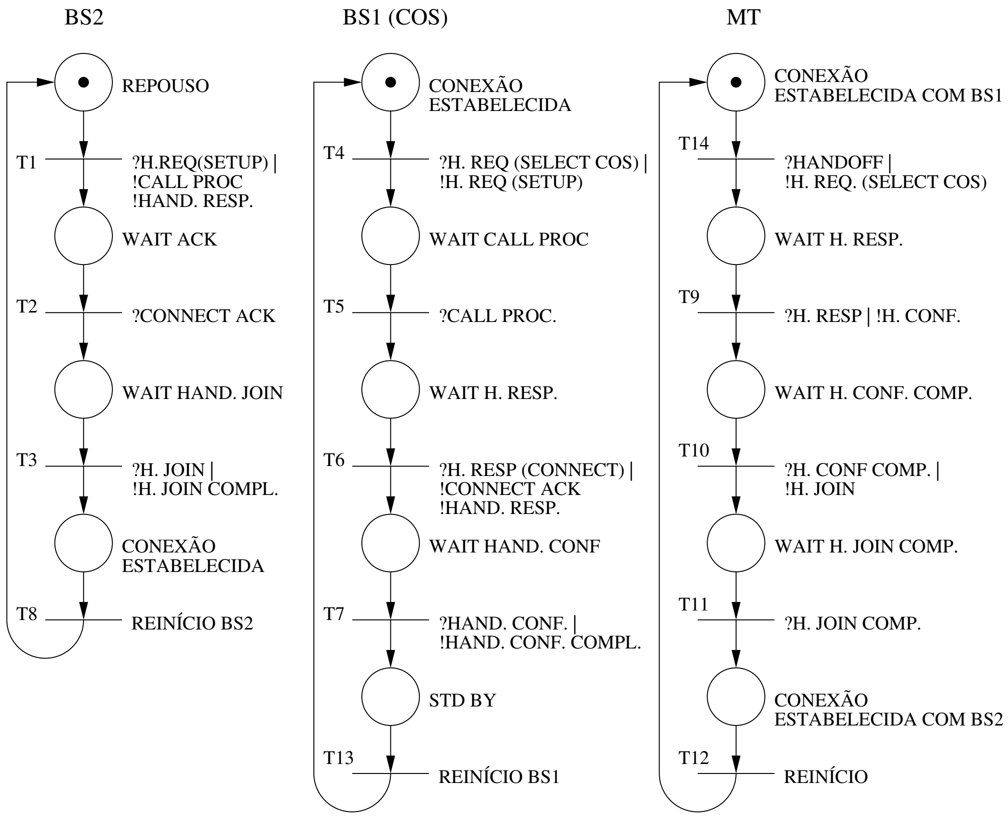

Figura 4.16: Máquina de Estados Finita do Protocolo de controle de *Handoff*.

Algumas Messagens	Função
HANDOFF REQUEST	Seleciona HOS/BS solicita novo caminho
HANDOFF RESPONSE	Conecta um novo caminho
HANDOFF CONFIRM	Informa ao HOS para habilitar um novo caminho
HANDOFF COMPLETE	Responde ao HANDOFF CONFIRM e libera o caminho antigo
HANDOFF JOIN	Completa a conexão de rádio entre a BS2 e o TM
HANDOFFJOIN-COMP	Conecta o link de radio

Tabela 4.1: Descrição das Primitivas de Serviço do Protocolo.

Aqui nos interessa impor restrições temporais importantes para o processo de *handoff*, mais especificamente, impor um limite de tempo para que processo se concretize. Portanto deve-se associar um valor de *delay* limite para as primitivas $HANDOFF\_REQ$ e $HANDOFF\_JOIN\_COMP$, que marcam respectivamente o início e o fim do processo. A especificação em Estelle Tempo-Real seria da seguinte forma:

```
NAME handoff;
FORALL t : time;
HENCEFORTH  (
SENDING_OF MCEP[2].H-REQ(SELECT_COS) AT t LEADSTO
(RECEPTION OF MCEP[2].H_JOIN_COMP)
AT (now <= t + 500));
```

Ou seja, o estabelecimento da conexão deve acontecer em um intervalo de no máximo 500 unidades de tempo. Neste caso o simulador apresentará ao usuário apenas o valor de tempo gasto no processo. Mecanismos como o de estabelecimento e encerramento de conexão nos obrigou a implementação de uma nova característica do simulador que é a monitoração de eventos e não só das PDUs, estes eventos são caracterizados por disparos de transições. a FI Tempo-Real que deve ser gerada pelo simulador é da seguinte forma:

```
[delay_de_trans]
transdeinicio THOFF
transdefim THJOINCOMP
corpodeinicio MT_BODY
corpodefim MT_BODY
modvardeinicio MT
modvardefim MT
[fim_delay_de_trans]
```

Onde $THOFF$ e $THJOINCOMP$ são os nomes dados, na especificação Estelle, às transições que marcam o início e o fim do evento. A partir do disparo da transição $THOFF$ o tempo começa a ser contado e termina com o disparo da transição de nome $THJOINCOMP$ e então tem-se o tempo total gasto para se efetuar o processo de *Handoff*

A seguir, é apresentado um trecho dos traços de simulação.

MT[0]	THOFF	0

```
|                            |----------------------|
|  --> MCEP[ 0]              | HREQMT               |
| MEDIUM[0]                  | MHREQ                | 0
|                            |----------------------|
|  --> MCEP1[ 0]             | HREQMED1             |
| BS1[0]                     | THREQ                | 0
|                            |----------------------|
|  --> MCEP[ 0]              | HREQBS1              |
| MEDIUM[0]                  | MHREQBS1             | 0
|                            |----------------------|
|  --> MCEP2[ 0]             | HREQMED2             |
| BS2[0]                     | THREQ2               | 0
|                            |----------------------|
```

Além disso vamos acompanhar trechos do andamento da simulação acompanhando os tempos em que os eventos ocorreram e o tempo para estabelecimento do *Handoff*. Considerando que a unidade de tempo utilizada foi milissegundos, e o valor do atraso entre eventos foi superior a 500, conforme mostrou a figura 4.17, portanto a restrição não foi cumprida. A especificação completa do protocolo de controle de *Handoff* para WATM pode ser encontrado apêndice B.

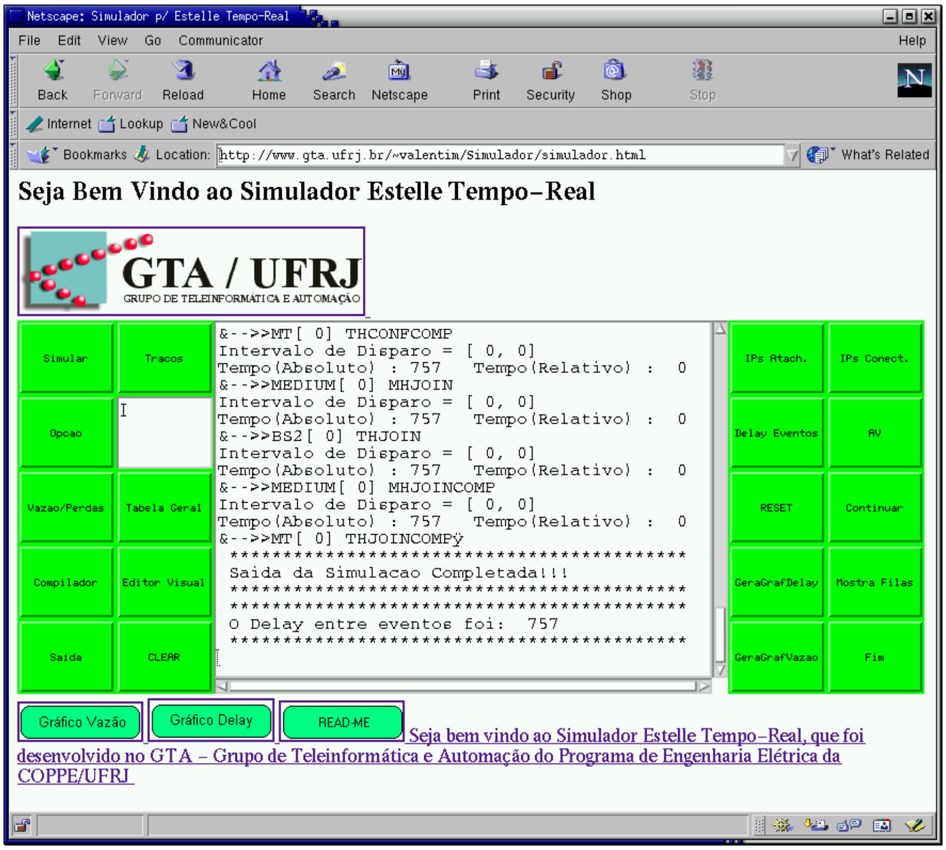

Figura 4.17: Saída do protocolo de controle de *Handoff*.

4.6 Conclusões

Este capítulo teve como objetivo apresentar os resultados de experimentos realizados com o simulador para Estelle Tempo-Real. Os modelos utilizados foram escolhidos de maneira que pudessem mostrar, ao máximo, as funcionalidades do simulador. Os modelos envolvendo fontes CBR visam retratar protocolos para transmissão de voz e vídeo, pois permitem checar os valores de atrasos pacote a pacote e de vazão instantânea. O protocolo para controle de Handoff permitiu verificar a medida de atraso entre eventos, no caso, foi medido o tempo de estabelecimento de uma nova conexão entre o terminal móvel e uma nova estação base.

Capítulo 5

Conclusões

A principal limitação da linguagem ESTELLE padrão diz respeito a sua incapacidade de especificar protocolos para aplicações multimídias, como voz e vídeo, que envolvem restrições temporais fim-a-fim, devido a característica da cláusula *Delay* que impõe restrição a uma transição de forma isolada. A extensão Estelle Tempo-Real traz operadores lógicos e palavras-chave que permitem ao projetista especificar sistemas distribuídos e protocolos que requerem tais restrições.

Este trabalho propôs um simulador para a linguagem Estelle Tempo-Real capaz de monitorar mensagens fim a fim, desde o seu envio no transmissor até a sua chegada no receptor, computando seus valores de *delay*, vazão e taxa de erros, parâmetros importantes na especificação de protocolos com restrições temporais.

Atrasos entre eventos também são calculados pelo simulador, esta característica permite especificar limites de tempo para fases de estabelecimento e encerramento de conexão, protocolos de controle de *Handoff*, tempo de resposta, etc.

O simulador conta com uma interface desenvolvida em JAVA e deve ser usada a partir de um *Browser* que mostra ao usuário os traços da simulação, fornecendo a seqüência das transições disparadas bem como o tempo de simulação em que elas foram disparadas, e gráficos de *delay* e *vazão*.

Uma limitação do simulador se faz presente quando é necessário que o usuário tenha acesso aos parâmetros da PDU para impor alguma condição temporal. Um alternativa seria acrescentar uma funcionalidade ao compilador que tornasse sempre disponível ao usuário os parâmetro da PDU, os valores de tempo total de simulação e os valores de tempo de disparo de cada transição individualmente. No que diz respeito ao cálculo da vazão, o simulador considera todos os pacotes tendo o mesmo tamanho, o que de fato acontece em mídias contínuas. O valor escolhido foi 1500 octetos porque é a MTU padrão para as redes *Ethernet*.

Ainda com relação ao cálculo da vazão instantânea, na tentativa de capturar as rajadas de tráfego que é a principal característica de mídias como voz e vídeo, há uma dificuldade em retratar este comportamento em uma especificação Estelle, onde o usuário não tem o controle sobre os tempos, dentro da faixa especificada, em que o emissor enviará pacotes.

Na tentativa de retratar de maneira fiel os sistemas reais, o usuário deve fazer uso dos recursos da linguagem para transpor para a especificação as situações que deverão ser enfrentadas. Os tempos atribuídos aos disparos de transição são hoje de inteira responsabilidade do usuário, com base na sua experiência.

Outro ponto onde pode haver outras contribuições é a Interface, enriquecendo o usuário de informações com exemplos. A criação de cadastros de usuários, onde seus exemplos pudessem ficar disponíveis a outros usuários na tentativa de disseminar informações.

Cursos a distância de projeto de protocolos também seria uma importante contribuição, o usuário teria acesso a um material didático e faria uso da ferramenta para testar seus exemplos e também poder obter informações de outros usuários mais experientes. Para isso deveria se usar tecnologias de páginas dinâmicas que tivessem acesso a banco de dados a partir da entrada do usuário. A utilização de PHP - Personal Home Pages e JSP - Java Server Pages são alternativas interessantes.

Referências Bibliográficas

[1] M. Diaz, *Formal Description Technique Estelle*. North-Holland, 1989.

[2] J.-P. Courtiat, *Formal Description Technique Lotos*. North-Holland, 1989.

[3] Estelle Development Toolset. Institut National des Telecommunications `http://www-lor.int-evry.fr /lor/edt/`.

[4] A. C. P. Pedroza, C. C. Goulart, P. R. O. Valim e R. de Oliveira, "Um Sistema de Auxílio ao Projeto de Protocolos de Comuniação para Redes de Computadores", in *Seminário Franco-Brasileiro de Sistemas de Informação Distribuídos (SFBSID'89)*, Florianópolis, SC, Brasil, 1989.

[5] J.-P. Courtiat e R. C. de Oliveira, "A reachability analysis of RT-Lotos specifications.", in 8^{th} *International Conference on Formal Description Techniques Protocol (Forte'95)*, Montreal, Canada, 1995.

[6] S. Fischer, "Real-Time Estelle", tech. rep., Universität Manheim, Germany, 1996.

[7] G. M. Delfino, J. V. dos Santos Filho, R. C. Roman, J. L. S. Leão e A. C. P. Pedroza, "An integrated toolset for the design of protocols with QoS requirements (ProtCAD/RT)", in *Protocols for Multimedia Systems - PROMS'2000 Conference*, Krakow, Poland, 2000.

[8] L. F. G. Soares, G. Lemos e S. Colcher, *Redes de Computadores - Das LANs MANs e WANs às Redes ATM*. Campus, 1995.

[9] J. F. Kurose e K. W. Ross, *Computer Networking*. Addison Wesley, 2001.

[10] S. Fischer, "On the Suitability of Estelle for Multimedia Systems", tech. rep., Universität Manheim, Germany, 1995.

[11] T. A. Henzinger, Z. Manna e A. Pnueli, “Timed Transition Systems”, *Real-Time: Theory in Practice (LNCS 600)*, 1991.

[12] R. Alur e T. A. Henzinger, “Logics and Models of Real Time : A Survey”, *Real-Time: Theory in Practice (LNCS 600)*, 1991.

[13] H. Bowman, G. Blair e A. Chetwynd, “Time versus Abstraction in Formal Descriptions”, *Formal Descriptions Techniques IV. Elsevier Science Publishers B.V. (North Holland)*, 1994.

[14] S. Leue, “Specifying Real-Time Requirements for SDL Specifications - a Temporal Logic-Based”, *Protocol Specification, Testing and Veryfication XV. Chapman & Hall*, 1995.

[15] S. Leue e S. Fischer, “Formal Methods for Broadband and Multimedia Systems”, *Computer Networks and ISDN Systems*, vol. 30, pp. 865–899, 1998.

[16] R. Gotzhein, “Temporal Logic and Applications - A Tutorial”, *Computer Networks and ISDN Systems*, vol. 24, pp. 203–218, 1992.

[17] J. S. Ostroff, “Temporal Logic of Real-Time Systems”, *Research Studies Press*, 1990.

[18] R. de Carvalho Roman, “Um editor visual para o auxílio a especificação de protocolos de comunicação com restrições temporais”, Tese de Mestrado, COPPE/PEE/UFRJ, junho 2002.

[19] R. C. de Oliveira Junior, “Um compilador para a linguagem Estelle”, Tese de Mestrado, COPPE/PEE/UFRJ, junho 1991.

[20] C. H. WEST, “Protocol Validation - Principles and Applications”, *Computer Networks and ISDN Systems*, 1992.

[21] O. B. Filho, *Verificação de Protocolos de Comunicação com Lógica Nebulosa*. Tese de Doutorado, COPPE/PEE/UFRJ, maio 1999.

[22] P. R. O. Valim, “Um simulador de protocolos de comunicação para a linguagem estelle”, Tese de Mestrado, COPPE/PEE/UFRJ, abril 1990.

[23] F. R. Loureiro, “Um simulador para a linguagem Estelle”, Tese de Mestrado, COPPE/PEE/UFRJ, maio 1995.

[24] P. M. Merlin e D. J. Farber, "Recoverability of Comunication Protocols - Implication of a Theoretical Study.", *IEEE Transactions on Communications*, vol. 24, pp. 1043–1046, setembro 1976.

[25] L. L. e Roger Cadenhead, *Aprenda em 21 dias - Java 1.2*. Campus, 1999.

[26] Gnuplot. `http://www.ucc.ie/gnuplot/gnuplot.html`.

[27] Universität Karlsruhe Manheim, Germany, *GMD Modula System Mocka*, 1999.

[28] W. R. Stevens, *UNIX Networking Programming - vol 1*. PH PTR, 1997.

[29] M. Yacoub, *Foundations of Mobile Radio Engineering*. CRC Press, 1993.

[30] A. B. A. Acharya, J.Li e D. Raychaudhuri, "Design and prototyping of location management and handoff protocols for wireless ATM networks", in *ICUPC'97*, Princeton, USA, 1997.

Apêndice A

Especificação completa do Sistema CBRRT

```
(* Especificação em Estelle Tempo Real do
sistema CBRRT - Produtor/Consumidor *)
Specification Sender_Receiver Systemactivity;
default individual queue; timescale miliseconds;
(* uma unica fila e' associada a cada ponto de interacao *)
{$ external NETUSER}
Type
data_type = record
n, nseq, te : integer;
                    end;
string = array [0..80] of char;
   procedure WriteLn; primitive;
```

```
   procedure WriteInt ( i : integer ); primitive;

   procedure WriteString ( s : string ); primitive;

Channel Sys_interface(i_user, i_provider);

by i_user:

START; STOP;

        Channel Medium_interface(i_user, i_provider);

by i_user:

DATREQ(m:data_type);

by i_provider:

DATIND(m:data_type);

  (* definicao dos modulos que compoem a especificacao *)

Module SendRecv activity;

  ip USR:Sys_interface(i_provider);

end;

body SendRecvBody for SendRecv;

module SenderType activity;

ip UCEP:Sys_interface(i_provider);

   MCEP:Medium_interface(i_user);

end;

body SenderBody for SenderType;

var

  m:data_type;

(* insercao dos parametros da interacao no InterfaceVariaveis *)
```

```
          {$afetar channel Medium_interface DATREQ := m.nseq}
state
  SenderIdle,
          DtTransfer;
initialize
to SenderIdle
name SenderStart:
  begin
    m.n:=0;
  end;
trans
          from SenderIdle
  when UCEP.START
          to DtTransfer
  name StartTransfer:
    begin
    end;
  from DtTransfer
  when UCEP.STOP
  priority 0
  to SenderIdle
  name EndTransfer:
    begin
```

```
        m.n:=0;
    end;
     from DtTransfer
  priority 1
  delay(30)
  to same
  name Tranfering:
  begin
    output MCEP.DATREQ(m);
    m.n:=m.n+1;
    if m.n>MAXSEQ
    then m.n:=0;
  end;
end;
module ReceiverType activity;
ip MCEP:Medium_interface(i_user);
end;
body ReceiverBody for ReceiverType;
var
  m:data_type;
(* insercao dos parametros da interacao no InterfaceVariaveis *)
          {$verificar channel Medium_interface DATIND m.nseq}
initialize
```

```
name RecvInit:
begin
end;
trans
  when MCEP.DATIND(m)
  name DtRecv:
          begin
           WriteString('Recebeu pacote ');
   WriteInt(m.n);
   WriteLn;
  end;
end;
  module MediumType activity;
ip MCEP: array [0..1] of Medium_interface(i_provider);
end;
body Medium_Body for Medium_Type;
var num,m:data_type;
state RcvData,
      SndData;
initialize
to RcvData
name MediumStart:
begin
```

```
end;
trans
       from RcvData
  when MCEP[0].DATREQ(m)
  to SndData
  name DtRcv:
  begin
   num.n := m.n;
   num.nseq := m.nseq;
   num.te := m.te;
  end;
  from SndData
  delay(50,200)
  to RcvData
  name DtDeliver:
  begin
    output MCEP[1].DATIND(num);
  end;
  end;
    modvar SENDER:SenderType;
               RECEIVER:ReceiverType;
               MEDIUM:MediumType;
       {SysSndRecv:SendRecv;}
```

```
initialize
name SndRcvStart:
  begin
    {init SysSndRecv with SendRecvBody;}
    init SENDER with SenderBody;
    init RECEIVER with ReceiverBody;
    init MEDIUM with MediumBody;
    attach USR to SENDER.UCEP;
    connect SENDER.MCEP to MEDIUM.MCEP[0];
    connect RECEIVER.MCEP to MEDIUM.MCEP[1];
  end;
end;
modvar SysSndRecv:SendRecv;
initialize
name SysInit:
begin
  init SysSndRecv with SendRecvBody;
end;
end.
```

Apêndice B

Especificação completa do Sistema MULTIPONTO-PONTO

```
Specification Multi_Sender_Receiver Systemactivity;

default individual queue; timescale miliseconds;

{$ external NETUSER}

  const

    MAXSEQ = 50;   {Numero de sequencia maximo a ser transmitido

- marca fim da transmissao}

    PKTSIZE = 2880;{Tamanho de pacotes de dados em bits}

    RATE = 96;{Taxa de transmissao em Kbps}

    MAXQ = 15;{Tamanho maximo da fila que simula perda no meio em

              numero de pacotes}

    MIN = 100;{Atraso minimo em ms, para transferencia fim a fim de

       dados}
```

```
    MAX = 150;{Atraso maximo em ms, no mesmo caso do anterior -
       ambos servem para calcular jitter}
    Type
     data_type = record
      n, nseq: integer;
                   end;
     string = array [0..80] of char;
     QType = array [0..MAXQ] of data_type;{Tipo de fila que simula
 a perda  no meio de transmissao}
(****Procedimentos de interface com o usuario da simulacao********)
   procedure WriteLn; primitive;
   procedure WriteInt ( i : integer ); primitive;
   procedure WriteString ( s : string ); primitive;
(*************Canal entre usuario e entidade emissora*************)
Channel Sys_interface(i_user, i_provider);
by i_user:
START; {Pedido do usuario para iniciar transmissao}
STOP;  {Pedido do usuario para finalizar transmissao}
(*************Canal entre emissor/receptor e meio***************)
        channel Medium_interface(i_user, i_provider);
by i_user:
DATREQ(m:data_type);
by i_provider:
```

```
DATIND(m:data_type);
 (*****Definicao dos modulos que compõem a especificacao ********)
(******Modulo mais externo que engloba emissor/receptor e meio***)
module SendRecv activity;
          ip USR: array [0..2] of Sys_interface(i_provider);
end;
body SendRecvBody for SendRecv;
  ip IUSR: array [0..2] of Sys_interface(i_user);
(***********Modulo Emissor**********************)
module SenderType activity;
ip UCEP:Sys_interface(i_provider);
   MCEP:Medium_interface(i_user);
end;
body SenderBody for SenderType;
var
  m:data_type;
 {$afetar channel Medium_interface DATREQ := m.nseq}
state
  SenderIdle,
          DtTransfer;
(********Inicialização do módulo emissor************)
initialize
to SenderIdle
```

```
name SenderStart:
          begin
    m.n:=0;
  end;
(*********Comportamento do módulo emissor************)
(***Pedido de usuário do sistema para iniciar a transmissão***)
trans
from SenderIdle
        when UCEP.START
        to DtTransfer
        name StartTransfer:
  begin
  end;
(*******Pedido de usuário para finalizar transmissão********)
from DtTransfer
        when UCEP.STOP
to SenderIdle
name EndTransfer:
  begin
      m.n:=0;
  end;
  (*******Dados sendo enviados a taxa constante**************)
from DtTransfer
```

```
delay(30)
to same
name Tranfering:
  begin
    output MCEP.DATREQ(m);
    m.n:=m.n+1;
  end;
(********Nada mais a transmitir*******************************)
from DtTransfer
provided m.n>MAXSEQ
to SenderIdle
name AllSent:
  begin
    m.n:=0;
  end;
(*******Restrições Temporais do módulo emissor - Taxa ********)
(* time constraints
  name TransferRate:
      forall t:time;
    forall seqnr:integer;
    henceforth(
    sending of MCEP.DATREQ[seqnr] at t
    leadsto sending of MCEP.DATREQ[seqnr+ 1000*RATE/PKTSIZE]
```

```
    at (now <t + 1000));
  end;
*)
end; {Corpo do emissor}
(*************Módulo Receptor***********************)
module ReceiverType activity;
ip MCEP:Medium_interface(i_user);
end;
body ReceiverBody for ReceiverType;
var
  num:data_type;
  {$verificar channel Medium_interface DATIND m.nseq}
(*******Inicialização do módulo receptor**********)
initialize
name RecvInit:
begin
end;
(**Comportamento do módulo receptor-apenas recebe dados******)
trans
when MCEP.DATIND(num)
name DtRecv:
        begin
        end;
```

```
end; {Corpo do emissor}
 (***********Módulo Meio de Transmissão************)
module MediumType activity;
ip MCEP: array [0..3] of Medium_interface(i_provider);
end;
body Medium_Body for Medium_Type;
var num,m:data_type;
    fifoq: QType;{Fila que simula perdas e atrasa pacotes para
  entrega}
    tail:integer;{Ponteiro para fim da fila}
    i: integer;
            Qempty: boolean;{Indica se fila esta vazia}
    Qfull: boolean;{Indica se fila esta cheia}
state RcvData,
      Qverify,
      SndData;
(********Inclusão de pacote na fila para transmissão*********)
Procedure PutQ(elem:data_type);
begin
          if not Qfull
          then begin
                fifoq[tail].n := elem.n;
    fifoq[tail].nseq := elem.nseq;
```

```
                WriteString('Pacote enfileirado =');
                WriteInt(elem.n);
                WriteLn;
                if tail = 0 then Qempty := false;
                if tail < MAXQ then tail := tail+1
                else Qfull := true;
               end
           else begin
                 WriteString ('Pacote ');
                 WriteInt(elem.n);
                 WriteString (' descartado.');
                 WriteLn;
                end;
end;
(*Retirada do primeiro pacote da fila para ser transmitido**)
Procedure GetFirst(var first:data_type);
var i: integer;
begin
  first.n := fifoq[0].n;
        first.nseq := fifoq[0].nseq;
        for i:= 0 to (tail-1)
  do begin
           fifoq[i].n := fifoq[i+1].n;
```

```
           fifoq[i].nseq := fifoq[i+1].nseq;
           end;
        fifoq[tail].n := -1;
        tail := tail - 1;
        if Qfull then Qfull := false;
        if tail = 0 then Qempty := true;
        end;
(**Inicialização do módulo meio de transmissão****)
initialize
to RcvData
name MediumStart:
  begin
    tail:=0;
    Qempty := true;
            Qfull := false;
            for i:=0 to MAXQ
    do fifoq[i].n:=-1;{Elemento=-1 marca lugar nao usado na fila}
end;
(*******Comportamento do módulo meio**************)
(*Chegam dados do emissor, meio enfileira e depois transmite*)
trans
        from RcvData
   when MCEP[0].DATREQ(m)
```

```
to SndData
name DtRcv:
  begin
    num.n := m.n;
            num.nseq :=m.nseq;
            PutQ(num);
        end;
   when MCEP[1].DATREQ(m)
to SndData
name DtRcv:
  begin
    num.n := m.n;
            num.nseq :=m.nseq;
            PutQ(num);
        end;
   when MCEP[2].DATREQ(m)
to SndData
name DtRcv:
  begin
    num.n := m.n;
            num.nseq :=m.nseq;
            PutQ(num);
        end;
```

```
(*******Novos dados chegando***********)

    from SndData

        when MCEP[0].DATREQ(m)

name MoreData:

  begin

    num.n := m.n;

            num.nseq := m.nseq;

            PutQ(num);

          end;

        when MCEP[1].DATREQ(m)

name MoreData:

  begin

    num.n := m.n;

            num.nseq := m.nseq;

            PutQ(num);

          end;

        when MCEP[2].DATREQ(m)

name MoreData:

  begin

    num.n := m.n;

            num.nseq := m.nseq;

            PutQ(num);

          end;
```

```
(******Encaminhamento de pacotes que estavam no meio******)
from SndData
        delay (100,200)
to Qverify
name DtDeliver:
  begin
    GetFirst(num);
            WriteString('Pacote retirado =');
    WriteInt(num.n);
            WriteLn;
    output MCEP[3].DATIND(num);
  end;
(******Nao há mais nada a ser transmitido*******)
from Qverify
provided Qempty
to RcvData
name verify:
  begin
  end;
(****Ainda h' pacotes esperando para serem enviados*****)
        provided not Qempty
to SndData
name morepkts:
```

```
  begin
  end;
  end; {Corpo do meio}
(*********Variáveis que representam os módulos**********)
modvar SENDER1,SENDER2,SENDER3:SenderType;
               RECEIVER:ReceiverType;
               MEDIUM:MediumType;
(*********Inicialização do módulo mais externo***********)
initialize
name SndRcvStart:
          begin
            init SENDER1 with SenderBody;
            init SENDER2 with SenderBody;
            init SENDER3 with SenderBody;
            init RECEIVER with ReceiverBody;
            init MEDIUM with MediumBody;
            connect IUSR[0] to SENDER1.UCEP;
            connect IUSR[1] to SENDER2.UCEP;
            connect IUSR[2] to SENDER3.UCEP;
            connect SENDER1.MCEP to MEDIUM.MCEP[0];
            connect SENDER2.MCEP to MEDIUM.MCEP[1];
            connect SENDER3.MCEP to MEDIUM.MCEP[2];
            connect RECEIVER.MCEP to MEDIUM.MCEP[3];
```

```
          end;
(*******Comportamento do modulo mais externo************)
trans
        when USR[0].START
    name STARTTRANS:
  begin
             output IUSR[0].START;
  end;
        when USR[0].STOP
    name STOPTRANS:
  begin
             output IUSR[0].STOP;
  end;
        when USR[1].START
    name STARTTRANS:
  begin
             output IUSR[1].START;
  end;
        when USR[1].STOP
    name STOPTRANS:
  begin
             output IUSR[1].STOP;
  end;
```

```
        when USR[2].START
    name STARTTRANS:
  begin
            output IUSR[2].START;
  end;
        when USR[2].STOP
    name STOPTRANS:
  begin
            output IUSR[2].STOP;
  end;
(**Restricões temporais do módulo mais exteno - jitter fim-a-fim***)
(* time constraints
  name jitter:
    forall t:time;
    forall seqnr: integer;
    henceforth (
    sending of SENDER.MCEP.DATREQ[seqnr] at t
    leadsto receiving of RECEIVER.MCEP.DATIND[seqnr]
    at (x+MIN<now<x+MAX));
  end;
end; {Corpo do modulo mais externo}
time constraints
  name jitter:
```

```
    forall t:time;
    forall seqnr: integer;
    henceforth (
    sending of SENDER.MCEP.DATREQ[seqnr] at t
    leadsto receiving of RECEIVER.MCEP.DATIND[seqnr]
    at (x+MIN<now<x+MAX));
  end;
*)
end; {Corpo do modulo mais externo}
(*********Variável módulo do sistema como um todo**************)
(********Inicialização da especificação*********************)
modvar SysSndRecv:SendRecv;
initialize
name SysInit:
begin
  init SysSndRecv with SendRecvBody;
end;
end.{Especficacao}
```

Apêndice C

Guia do Usuário

Antes de iniciar o processo de simulação, leia atentamente às instruções baixo. Para dar início ao processo de simulação, faça:

i) Click no Botão Simular

ii) Selecione a opção (1), digitando o valor na pequena caixa de texto e em seguida "clicando" em opção. Esta opção permite ao usuário introduzir a interação que dará início à execução da máquina de estados;

iii) Selecione (como descrito no ítem 1) o módulo responsável pelo início da execução da máquina de estados;

iv) Selecione o ponto de interação onde será depositada a interação de iniciará o processo;

v) Selecione a interação que dará início ao processo de simulação;

vi) Se você desejar iniciar mais algum módulo (no caso de haver mais de um transmissor), click em "Simular", senão, click em "Continuar";

C.1 Descrição das Funcionalidades

i) Botão "*Simular*" : Inicia o processo de simulação;

ii) Botão "*Traços*" : Mostra os traços da simulação;

iii) Botão "*Opção*" : Seleciona a opcão desejada. Deve ser pressionado após o valor ser digitado na caixa de texto;

iv) Botão "*Vazão/Taxa de Perdas*" : Mostra a vazão média e taxa de perdas calculada;

v) Botão "*Tabela Geral*" : Mostra os resultados gerais;

vi) Botão "*Compilador*" : Interface com o compilador está prevista, mas não está implementada;

vii) Botão "*Editor Visual*" : Interface com o editor visual está prevista, mas não está implementada;

viii) Botão "*Saída*" : Disponibiliza a saída da simulação para o usuário;

ix) Botão "*Clear*" : Limpa a tela;

x) Botão "*IPs Atach.*" : Mostra os pontos de interação atachados;

xi) Botão "*IPs Conect.*." : Mostra os pontos de interação conectados;

xii) Botão "*Delay Eventos*" : Mostra o valor calculado para o *delay* de eventos;

xiii) Botão "*RESET*" : *Reseta* os dados de uma simulação permitindo o início de uma nova sessão;

xiv) Botão "*Continuar*" : Permite continuar uma simulação ao invés de iníciar um novo módulo;

xv) Botão "*GeraGrafDelay*" : Gera o gráfico do *delay*. Esta opção deve ser selecionada antes de clicar no botão "Gráfico Delay";

xvi) Botão “*GeraGrafVazão*” : Gera o gráfico da vazão. Esta opção deve ser selecionada antes de clicar no botão “Gráfico Vazão”;

xvii) Botão “*Mostra Filas*” : Mostra o conteúdo das filas associadas aos pontos de interação;

xviii) Botão “*FIM*” : Finaliza a simulação.