Capítulo III

Especificação Formal e Validação

O crescente aumento no uso de comunicação de dados, seja em redes de computadores ou em automação de processos industriais, gerou a necessidade de criação de técnicas de descrição formal de protocolos que, além de especificar de forma clara, concisa e sem ambigüidades, permitem a verificação, a simulação, e até a obtenção de uma implementação de forma automática a partir da especificação. Em implementações não automáticas (o caso deste trabalho) seria ainda possível determinar se a implementação está de acordo com a especificação.

Dentre estas Técnicas de Especificação Formal (FDT - *Formal Description Techniques*), destacam-se as normas internacionais da ISO: **ESTELLE** (*Extended State Transition Language*) [11] e **LOTOS** (*Language of Temporal Ordering Specifications*) [12], que são dois formalismos usados para descrições formais de sistemas distribuídos e, em particular, de protocolos de comunicação, assim como dos serviços oferecidos por um protocolo.

Neste trabalho foi usado o formalismo definido pela linguagem ESTELLE. Esta escolha se deve à fácil modelagem da máquina de estados finita definida no capítulo II, aliada à disponibilidade de ferramentas para ESTELLE desenvolvidas no GTA da UFRJ.

As ferramentas desenvolvidas [13] permitem desde um auxílio à edição da especificação até a validação com o uso de seqüências de testes geradas de forma automática. São elas:

- Editor de ESTELLE: permite a edição orientada de especificações ESTELLE, inclusive com recursos gráficos [14];

- Compilador [15] e Simulador ESTELLE [16]: estes dois módulos operam de forma integrada, permitindo a compilação de uma especificação ESTELLE para uma Forma Intermediária (FI) em linguagem módula 2, que pode ser executada com intervenção do usuário. Existe ainda a geração de um relatório de acompanhamento (*Trace*) para o usuário, do qual constam cada transição disparada, na ordem cronológica em que

ocorreu. O apêndice B mostra um relatório gerado durante uma curta sessão de simulação, em que foi realizada uma conexão multiponto-a-multiponto, a transmissão de uma mensagem pela estação que pediu a conexão, o pedido da ficha por uma outra estação, o envio da ficha, a transmissão de uma mensagem pela nova estação Mestra e, em seguida a desconexão multiponto-a-multiponto;

- Gerador de Seqüência de Testes [17]: permite a geração automática de seqüências de teste a partir da especificação dos estados e das transições de uma máquina de estados finita. Este gerador foi utilizado para gerar seqüências de primitivas do usuário do serviço proposto neste trabalho. De posse das seqüências geradas, estas foram colocadas como entradas no simulador ESTELLE.

Neste capítulo será dada uma breve apresentação da linguagem ESTELLE, seguida da especificação realizada, dos testes e dos resultados obtidos.

III.1. A Linguagem ESTELLE

Tendo sido desenvolvida pela *International Standards Organization* (ISO), a linguagem ESTELLE é uma técnica de descrição formal baseada em um modelo de transição de máquina de estados finita estendida, e escrita usado instruções da linguagem Pascal e algumas construções modulares de Ada. Para se especificar um protocolo em ESTELLE, é necessário a definição das **entidades** cooperantes que irão interagir entre si, através da troca de informações por **canais de comunicação**.

Em ESTELLE cada modelo de entidade é definida em um **módulo**. Cada módulo é definido pela associação de um cabeçalho (*header*), que caracteriza a sua visibilidade externa, e por um corpo (*body*) que caracteriza o comportamento de cada **instância** de módulo. Um módulo é uma definição abstrata (um modelo) que pode ser utilizada para criação de várias instâncias deste mesmo módulo, durante a simulação de uma especificação ESTELLE. Uma instância de um módulo possui nome diferente deste e das demais instâncias, sendo o nome dado após a execução da primitiva ESTELLE **"init"**. As instâncias criadas a partir de um mesmo módulo representam as entidades pares em uma simulação ESTELLE. Além disto, é possível a definição de um módulo (ou mais) aninhado dentro de outro módulo, caso em que o simbolismo filho e pai é usado para referenciar o primeiro ao segundo. Desta forma, módulos filhos definidos dentro de um mesmo módulo pai são chamados de irmãos, entre si.

Um canal de comunicação ESTELLE não deve ser confundido com o canal de comunicação usado pelo protocolo, que pode ser definido pelo provedor do serviço. Um canal de comunicação (meio de transmissão) entre duas entidades pares do protocolo é também

definido como uma entidade (através da definição de um módulo com as características deste meio de comunicação). Um canal ESTELLE transmite, de forma bidirecional, informação entre dois módulos conectados. Estes dois módulos, que são descrições abstratas de duas entidades (por exemplo, usuário e provedor), tem sua conexão formalizada pela definição de um **tipo de canal**, que especifica um conjunto de primitivas de interação entre as duas entidades. Esta definição contém uma lista de **pontos de interação** através dos quais o módulo interage com seu exterior e é também abstrata, Nesta definição é utilizado o conceito de papel assumido por cada módulo conectado a cada extremo do canal (por exemplo: usuário e provedor). Através deste papel, determina-se quais são as interações que um módulo pode enviar, dentre aquelas declaradas pelo canal. A informação é transmitida entre instâncias de um módulo através de **parâmetros** da primitiva de interação.

Na definição do cabeçalho de um módulo determina-se, através de atributos, se o módulo é: totalmente assíncrono (independente) das demais instâncias de módulos (*System*) ou cooperante (sem atributo *System*) e; se a execução dos módulos será síncrona paralela, com disparo simultâneo de todas as transições sensibilizadas em um mesmo "instantâneo", desde que não estejam em conflito pai/filho (atributo *Process*) ou, se a execução será não determinística, preservada a prioridade entre pais e filhos, com o disparo de somente uma das transições sensibilizadas em um determinado "instantâneo", transição esta escolhida de maneira não determinística (atributo *Activity*). A determinação da visibilidade externa é feita através da definição dos pontos de interação e do conjunto de variáveis exportadas pelo módulo. Os parâmetros passados no momento da criação de uma instância de módulo e que são utilizadas em seu corpo, também são definidas no cabeçalho.

Já a definição de um corpo de módulo descreve o comportamento interno de uma instância de módulo, em termos de um modelo de transição. A definição de um corpo de módulo pode ser dividida em quatro partes, a saber:

- declarações: podem ser declaradas constantes, tipos, cabeçalhos de módulos filhos, pontos de interação internos, procedimentos, funções e, variáveis locais.

- iniciação: declaração de uma transição (ou um conjunto de transições) que será executada no momento da iniciação da instância do módulo. No caso de existir um conjunto de transições, somente uma delas seja disparada no momento da iniciação.

- transições: declaração das transições que descrevem a máquina de estados finita que corresponde ao comportamento deste corpo de módulo.

- terminação: quando uma instância deste corpo de módulo é destruída, o procedimento aqui declarado é executado.

As transições que especificam o comportamento do corpo de um módulo são definidas através de **cláusulas** ESTELLE. Elas determinam as condições necessárias a serem satisfeitas para o disparo da transição e as ações que ocorrem quando o disparo acontece. Os seis tipos de cláusulas e suas funções são descritos a seguir:

FROM <estado de entrada>: indicar o estado antes do disparo;

TO <estado de saída>: indicar o estado após a saída;

WHEN <interação de entrada>: receber interações assíncronas que ficam depositadas em filas tipo FIFO;

PROVIDED <expressão booleana>: estabelecer uma condição a ser satisfeita pelas variáveis Pascal da instância do módulo;

PRIORITY <constante inteira>: introduzir uma prioridade de disparo entre transições;

DELAY <tempo>: estabelecer um tempo mínimo e máximo, entre os quais deve disparar uma transição que tenha sido sensibilizada, isto é, que satisfaz às cláusulas anteriores.

Além destas cláusulas que determinam a sensibilização ou não de uma determinada transição, existe a cláusula *NAME* . Esta não influencia na sensibilização de uma transição e sua função é dar um nome, escolhido pelo usuário do simulador ESTELLE, à transição onde a cláusula aparece. Este nome é usado na fase de simulação, sendo colocado no relatório de acompanhamento (*Trace*).

Quando uma transição é disparada, podem existir comandos da linguagem Pascal que são executados e chamadas a primitivas ESTELLE. Isto permite a criação e/ou destruição de instâncias de módulos, estabelecimento de conexão/desconexão (ESTELLE) entre módulos e o envio de interações assíncronas através dos pontos de interação do módulo.

No compilador desenvolvido em [15] foram introduzidas diretivas que não fazem parte da especificação ESTELLE diretamente mas, permitem a introdução de outras extensões à linguagem ou de comodidades para o usuário, como: acompanhamento de variáveis; parâmetros e interações durante a execução de uma simulação passo a passo de uma especificação; a introdução de heurísticas que auxiliem na verificação formal de propriedades e a localização de procedimentos ou corpos de módulos declarados como *external*. Estas diretivas utilizam o símbolo de comentário em ESTELLE, as chaves, onde o caracter imediatamente sucessor do "abre chaves" deve ser o "**$**". Uma diretiva possui o seguinte formato:

{$ nome_da_diretiva parâmetros }

III.2. Aspectos da formalização em ESTELLE do Protocolo Multiponto-a-Multiponto

Nesta seção é feita uma descrição resumida da especificação do serviço proposto em ESTELLE, seguindo a ordem da descrição do Apêndice A. Também são descritos os problemas encontrados e soluções adotadas.

III.2.1. Arquitetura da especificação

A arquitetura adotada para a especificação é mostrada na Figura III.1. Esta arquitetura modela a troca de dados entre quatro usuários da camada de enlace (UA a UD). A troca de dados é controlada por quatro entidades Multiponto-a-Multiponto (ENL_MPMPa a ENL_MPMPd), conectadas através do meio de comunicação MACPHY.

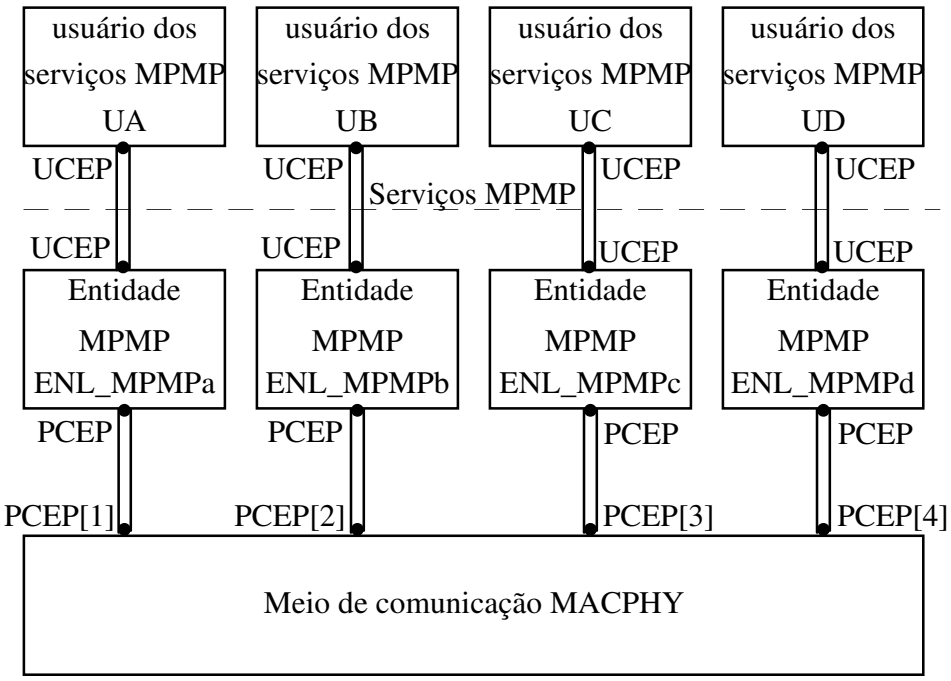

Figura III.1 - Arquitetura da especificação do protocolo Multiponto-a-Multiponto.

III.2.2. Descrição da especificação

Neste trabalho utilizou-se um único módulo tipo *System Activity*, dentro do qual foram definidos módulos filhos tipo *Activity*. No módulo principal (*System Activity*) foram declarados, na parte de declarações, constantes (número máximo de estações participantes, o valor inicial atribuído a temporizadores), tipos (nome de PDUs, formato - estrutura - utilizado em PDUs, a estrutura utilizada na Fila de Espera e na Lista de Participantes), além da

declaração de funções de escrita na tela e leitura de teclado, definidas como primitivas. Esta declaração de novas primitivas foi possível com uso da diretiva {$ external netuser }, que possibilita o uso destas funções definidas externamente, como extensões da linguagem ESTELLE.

Foi definido um único formato de PDU. Este formato declarado teve sua estrutura definida de modo a suportar todos os requisitos das PDUs utilizadas pelo protocolo, as quais foram definidas no capítulo II. Esta solução foi adotada para facilitar a montagem, o envio e o recebimento das PDUs, em detrimento da otimização da utilização do meio de comunicação, já que esta não é uma preocupação nesta especificação em ESTELLE. Na montagem de uma PDU, os campos da estrutura que não são utilizados permanecem sem sofrer alterações.

Em seguida são declarados os dois Tipos de Canais utilizados:

- canal MPENL_USU_Interface - usado para conexão entre instâncias de módulos do usuário (UA a UD) e instâncias de módulos do tipo Entidade_MPENL (ENL_MPMPa a ENL_MPMPd). Os módulos de usuário não foram implementadas nesta especificação. O canal MPENL_USU_Interface foi utilizado como interface com o usuário (humano) do simulador. Os testes e simulações são feitos inserindo-se as primitivas de serviço diretamente no canal. As primitivas recebidas pelo usuário são sinalizadas através de mensagens na tela de vídeo.

- canal MPENL_PH_Interface - usado para conexão entre instâncias do tipo Entidade_MPENL (ENL_MPMPa a ENL_MPMPd) e a instância do tipo Tipo_MACPHY (MACPHY).

A seguir tem-se as declarações dos dois Tipos de Canais em ESTELLE:

```
CHANNEL MPENL_USU_Interface (usu, enlace);
 BY usu:
   MPDL_EST;                { MPDL_ESTABLISH.Request }
   MPDT(sdu:Tipo_Dado);       { MPDL_DATA.Request }
   MPDL_AB;                 { MPDL_ABORT.Request }
   MPDL_RL;                 { MPDL_RELEASE_PLEASE.Request }
   MPDL_IND_AB(ID:Integer);    { MPDL_INDIVIDUAL_ABORT.Request }
   MPDL_IND_RL;    { MPDL_INDIVIDUAL_RELEASE_PLEASE.Request }
   MPDL_RY(ID:Integer);      { MPDL_REPLY.Request }
   MPDL_TG;                 { MPDL_TOKEN_GIVE.Request }
   MPDL_TP;                 { MPDL_TOKEN_PLEASE.Request }
   DERRUBA;                 { coloca a estação em CLOSED }

 BY enlace:
   MPDL_EST_Ind(fonte:integer);   { MPDL_ESTABLISH.Indication }
```

```
MPDL_EST_Conf(stat:tipo_conf); { MPDL_ESTABLISH.Confirm }
MPDL_AB_Ind(razao:tipo_razao); { MPDL_ABORT.Indication }
MPDL_AB_Conf;                  { MPDL_ABORT.Confirm }
MPDL_RL_Ind;                   { MPDL_RELEASE_PLEASE.Indication }
MPDL_DT_Ind(sdu:Tipo_Dado);    { MPDL_DATA.Indication }
MPDL_IND_AB_Ind;               { MPDL_INDIVIDUAL_ABORT.Indication }
MPDL_IND_AB_Conf(id:integer);
                  { MPDL_INDIVIDUAL_ABORT.Confirm }
MPDL_IND_RL_Ind(id:integer);
                  {MPDL_INDIVIDUAL_RELEASE_PLEASE.Indication }
MPDL_TP_Ind(id:integer);       { MPDL_TOKEN_PLEASE.Indication }
MPDL_TG_Ind;                   { MPDL_TOKEN_GIVE.Indication }
STOP_TX;                       { MPDL_STOP_TRANSMISSION.Indication }
```

```
CHANNEL MPENL_PH_Interface (enlace, phy);
 BY enlace:
  DT_Req(PDU:Tipo_PDU);
 BY phy:
  DT_Ind(PDU:Tipo_PDU);
```

Vale frisar a inserção de uma primitiva não constante da especificação do serviço: a primitiva DERRUBA. Ela foi inserida com a finalidade de tirar uma estação da conexão abruptamente, simulando um defeito grave ou uma eventual falta de luz. Esta primitiva permitiu avaliar o protocolo nestes casos de falha grave.

A especificação contêm dois tipos de módulos: Entidade_MPENL e Tipo_MACPHY. As relações entre diferentes tipos de módulos e canais com a arquitetura da especificação é mostrada na Figura III.2, onde estão representadas apenas duas estações ao invés de quatro, para simplificar a figura.

O cabeçalho de módulo usado na definição da instância da entidade que presta o serviço proposto é então declarado e, em seguida, o corpo desta mesma entidade. Este cabeçalho é mostrado a seguir:

```
MODULE Entidade_MPENL activity (aux_id_est:Integer);

IP UCEP : MPENL_USU_Interface (enlace);
    PCEP : MPENL_PH_Interface (enlace);

END;
```

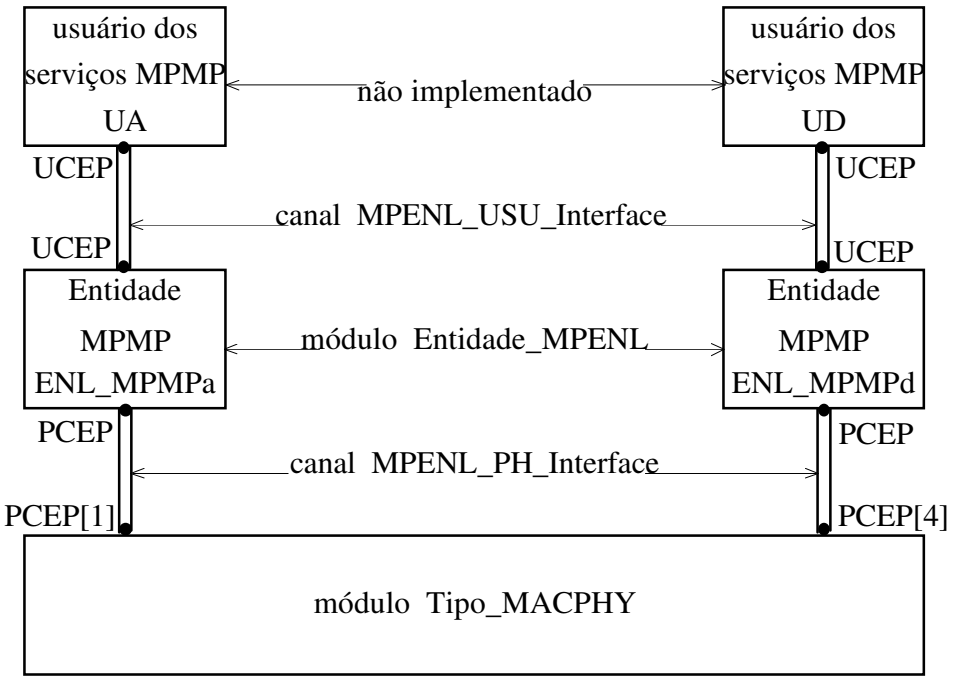

Figura III.2 - Relação entre a arquitetura da especificação e os tipos de módulos e canais utilizados.

O corpo de módulo da entidade que presta o serviço proposto possui a especificação, em ESTELLE, da máquina de estados do serviço, sendo a parte principal desta especificação. Nele são inicialmente declaradas as variáveis locais de uma instância, seguida de diretivas {$ verificar nome_var }, que permitem a verificação da variável mencionada em cada diretiva, durante a execução. Os nomes dos estados existentes são declarados em seguida (*STATE*).

Por ser muito extenso, este corpo não será totalmente reproduzido aqui estando, porém, completo no Apêndice A. No entanto, alguns trechos mecerem ser comentados.

Os temporizadores utilizados necessitam ser criados ou finalizados a qualquer tempo. Por isso não se usou somente a cláusula *DELAY*. Para "implementá-los" em ESTELLE foi usada uma variável de controle para cada um, que é iniciada com o valor de contagem. Uma transição com uma cláusula *DELAY* decrementa o valor desta variável até que seu valor seja menor que 0 (-1). Este valor (-1) indica que o temporizador está inativo. A cada transição desta, o valor do temporizador é testado e, quando for zero, o contador é dado como "estourado". Para finalizar um temporizador basta atribuir à sua variável de controle o valor -1 (na verdade, qualquer valor menor que zero). O techo a seguir em ESTELLE mostra a gerência do contador T_R, durante a espera da confirmação de desconexão multiponto-a-multiponto de todas as estações. O procedimento Msg_Id("msg") coloca na tela a mensagem especificada no parâmetro, juntamente com a identificação da estação (instância) que gerou a

ocorrência. A variável "to_closed" é utilizada para sensibilizar a transição que irá desconectar a estação. O trecho em que esta transição ocorre também é mostrado em seguida:

```
Trans { timer }
 From AWAITING_RELEASE
   to same
   delay(1)
   name AWRELTIMER:
     Begin
     if cont_tr>=0 then cont_tr:=cont_tr-1;
     if cont_tr=0 Then  (* estourou TR *)
       Begin
       Msg_Id(': estourou TR em AWAITING_RELEASE.');
       if num_tent>MAX_TENT
         Then Begin  (* P2 *)
             writestring(' Envia MPDL_ABORT.Conf.');
             to_closed:=true
             End
        Else Begin    (* NP2 *)
             writestring(' Difundindo novo pedido de desconexao.');
             Cria_MPDISC(pdu_aux,id_est);
             Output PCEP.DT_Req(pdu_aux);
             cont_tr:=TR;
             num_tent:=num_tent+1  (* A7 *)
             End
       End; { estourou TR }
     Atualiza_Verifica_Trq
     End;
```

```
From AWAITING_RELEASE
  Provided to_closed
  to INICIA
  name AWRELCLOSED:
   Begin
   writeint(id_est);
   to_closed:=false;
   writestring(': fechou')
   End;
```

Também merece destaque o fato de que a maioria das rotinas de manipulação das listas e filas escritas, puderam ser literalmente traduzidas para linguagem C, durante a etapa de implementação (Capítulo IV). Isto certamente poupou tempo, pois algumas das rotinas já estavam funcionando e exaustivamente testadas.

Uma entidade MAC-PHY foi idealizada, assumindo-se que seus serviços estariam sempre disponíveis. Seu cabeçalho e corpo são declarados, um após o outro. O cabeçalho especifica

uma comunicação apenas com o usuário do serviço contendo, no entanto, um número de pontos de interação que depende da constante que especifica o número máximo de estações (entidades) participantes. Este cabeçalho está reproduzido abaixo:

```
MODULE Tipo_MACPHY activity ;
 IP
 PCEP : Array[1..NUM_ESTACOES] Of MPENL_PH_Interface(phy);
END;
```

O corpo do módulo MACPHY foi inicialmente especificado sem introduzir perda de quadros. A sensibilização de suas transições dependem do ponto de entrada da interação e do endereço de destino da PDU. Uma PDU pode ser endereçada a uma única estação, caso em que a PDU é enviada somente ao ponto de interação desta estação ou, ser uma difusão, quando a PDU é enviada a todos os demais pontos de interação. Este trecho da especificação é reproduzido a seguir.

```
BODY CORPO_MACPHY FOR Tipo_MACPHY;
 TRANS
   WHEN PCEP[1].DT_Req
    Provided pdu.destino=255
    name DT_1_all:
     BEGIN
     OUTPUT PCEP[2].DT_Ind(PDU);
     OUTPUT PCEP[3].DT_Ind(PDU);
     OUTPUT PCEP[4].DT_Ind(PDU);
     END;
    Provided Otherwise
     name DT_1:
     Begin
     Output PCEP[pdu.destino].DT_Ind(PDU)
     End;
   WHEN PCEP[2].DT_Req
    Provided pdu.destino=255
    name DT_2_all:
     BEGIN
     OUTPUT PCEP[1].DT_Ind(PDU);
     OUTPUT PCEP[3].DT_Ind(PDU);
     OUTPUT PCEP[4].DT_Ind(PDU);
     END;
    Provided Otherwise
     name DT_2:
     Begin
     Output PCEP[pdu.destino].DT_Ind(PDU)
     End;
```

```
  WHEN PCEP[3].DT_Req
   Provided pdu.destino=255
   name DT_3_all:
    BEGIN
    OUTPUT PCEP[1].DT_Ind(PDU);
    OUTPUT PCEP[2].DT_Ind(PDU);
    OUTPUT PCEP[4].DT_Ind(PDU);
    END;
   Provided Otherwise
    name DT_3:
    Begin
    Output PCEP[pdu.destino].DT_Ind(PDU)
    End;
  WHEN PCEP[4].DT_Req
   Provided pdu.destino=255
   name DT_4_ALL:
    BEGIN
    OUTPUT PCEP[1].DT_Ind(PDU);
    OUTPUT PCEP[2].DT_Ind(PDU)
    OUTPUT PCEP[3].DT_Ind(PDU);
    END;
   Provided Otherwise
    name DT_4:
    Begin
    Output PCEP[pdu.destino].DT_Ind(PDU)
    End;

END;
```

A especificação em ESTELLE deste trabalho foi iniciada em microcomputadores IBM-PC compatíveis. O método de trabalho utilizado foi o de definição gradual das transições essenciais, testando-as imediatamente para, em seguida, inserir as transições que tratavam exceções. No entanto, já com a especificação em estado adiantado, foi verificado que o compilador não permite a geração de códigos maior do que 64 KBytes neste ambiente de trabalho (micros IBM-PC compatíveis). Isto levou a adoção de medidas de economia no tamanho do código. A principal medida foi a utilização extensiva de procedimentos (subrotinas), contendo pedaços de códigos que se repetiam na especificação. Isto explica a quantidade de procedimentos que aparecem no inicio da especificação do corpo de módulo da entidade que presta o serviço. Mas, mesmo assim, não foi possível a inclusão de toda a máquina de estados, por causa da limitação no tamanho do código. Neste ponto do trabalho grande parte da especificação já estava pronta, e já era possível a conexão multiponto-a-multiponto, o envio de mensagens e a desconexão multiponto-a-multiponto, assim como serviços de pedido de ficha, pedido de desconexão, desconexão individual. Enfim, faltavam apenas transições de tratamento de algumas das exceções previstas. Com a migração do

compilador/simulador ESTELLE para o ambiente UNIX, podendo rodar em estações SUN, foi possível migrar a especificação do serviço proposto e terminar a especificação/simulação em ESTELLE.

III.3. Testes com geração automática de seqüências

O Gerador de seqüência de testes foi utilizado para simular todas as possíveis situações. O Gerador de seqüência de teste foi alimentado com uma máquina de estados criada a partir do serviço multiponto-a-multiponto (primitivas de serviço). Como resultado, foi gerado um relatório com diversas seqüências de envio/recebimento de primitivas de serviço. As seqüências de testes obtidas no Gerador foram inseridas, uma a uma, no simulador ESTELLE.

Através deste procedimento foi detectada a falta de duas transições existentes na máquina de estados do protocolo (Capítulo II), mas que não foram especificadas em ESTELLE. Estas transições são as que manipulam as primitivas MPDL_RELEASE_PLEASE.Request / MPDL_RELEASE_PLEASE.Indication e MPDL_INDIVIDUAL_RELEASE_PLEASE.Request / MPDL_INDIVIDUAL_RELEASE_PLEASE.Indication. A Figura III.3 mostra as duas seqüências não especificadas.

Estas duas seqüências de primitivas permitem a uma estação em estado de recepção pedir a liberação negociada à estação que for a Mestra no momento. Nos dois casos, a única função do protocolo é transferir a PDU de pedido e, na estação mestra, enviar a primitiva (Indication). A negociação é feita a nível de aplicação, e não pelo serviço.

A seguir é mostrada as duas seqüências de teste que detectaram a falha:

```
10 = r/N,MPDL_Establish_Req/MPDL_Establish_Conf,
     MPDL_Individual_Release_Please_Req/Null, Null/MPDL_Abort_Ind
11 = r/N,MPDL_Establish_Req/MPDL_Establish_Conf,
     MPDL_Release_Please_Req/Null, Null/MPDL_Abort_Ind
```

Após a inclusão das duas transições o protocolo passou nos testes.

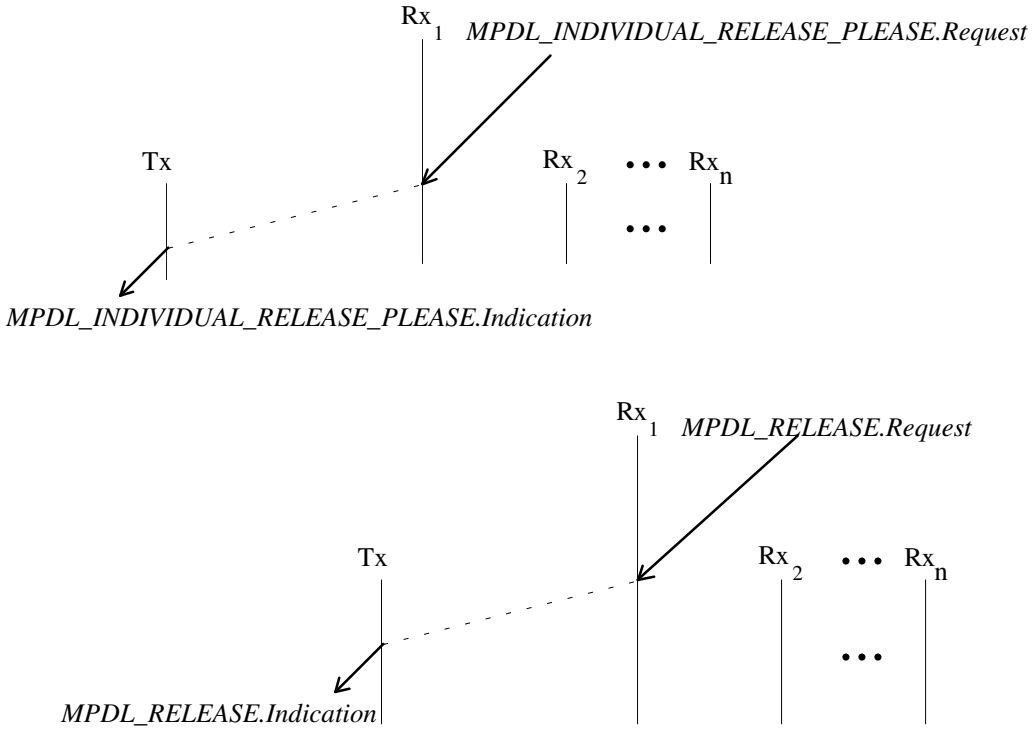

Figura III.3 Seqüências de primitivas não especificadas em ESTELLE que foram detectadas pela seqüência de testes.

A listagem completa com as seqüências de testes obtidas no Gerador e usadas no Simulador ESTELLE encontra-se no Apêndice C.

III.4. Comentários

A especificação de um protocolo com Técnicas de Descrição Formal torna o protocolo livre das ambigüidades decorrentes da utilização de linguagem natural na especificação apresentada no Capítulo II. Uma vez que o sistema tenha sido descrito usando uma FDT, sua implementação torna-se única e universal. A importância desta abordagem cresce mais ainda quando da concepção de um protocolo de comunicação novo, em sua primeira versão, como o apresentado neste trabalho.

O protocolo apresentado neste trabalho teve sua máquina de estados totalmente especificada e simulada em ESTELLE, com o auxílio das ferramentas para CAD ESTELLE desenvolvidas pelo GTA da UFRJ, descritas neste Capítulo.

Alguns artifícios de teste como transições que perdem quadros na entidade que serve de meio de comunicação (MACPHY), foram inseridos para simular condições de erro. Realizando interações manuais com seqüências aleatórias de primitivas (sem uso do gerador de seqüência

de teste), foram possíveis correções de alguns erros relacionados ao tratamento de exceções, como queda de estações durante a fase de transmissão de dados e, com isso, a inclusão de transições não previstas inicialmente, mas necessárias.

O uso das seqüências de teste apontou ainda uma falha na especificação em ESTELLE: duas transições existentes na máquina de estados do protocolo não haviam sido especificadas em ESTELLE.

A simulação no ambiente CAD Estelle apontou alguns problemas no protocolo, principalmente no que tange ao tratamento de exceções. Isto mostrou o quanto é difícil conceber um protocolo com esta complexidade, sem erros. A utilização das ferramentas aqui apresentadas é um dos caminhos essenciais para a validação de um protocolo.