4º SBRC
RECIFE - 24 A 26 DE MARÇO 86

CONSIDERAÇÕES SOBRE A UTILIZAÇÃO DE TÉCNICAS DE DESCRIÇÃO FORMAL DE PROTOCOLOS

Aroldo Y. Yai, DIGIREDE Informática LTDA.

Jaime Ono, Lynx Tecnologia LTDA.

Lauro T. Muramoto, EPUSP, FDTE.

William Astolfi, Scopus Tecnologia S.A.

Stefania Stiubiener, Escola Politécnica da USP
End. Cidade Universitária "Armando Salles de Oliveira"
Caixa Postal 11.455 - São Paulo - SP.

RESUMO

Neste Artigo são feitas algumas considerações sobre a utilização de Técnicas de Descrição Formal de Protocolos pela ISO e sobre a utilização de Redes de Petri para esse fim. Os exemplos escolhidos tem como base o Protocolo de Transporte Classe 2 da ISO.

1. Introdução

O desenvolvimento da computação distribuida, principalmente o aspecto de projeto, implementação e operação dos programas de comunicação de mensagens num ambiente de sistemas abertos, conduziu à preocupação em relação à especificação formal de protocolos.

Como expressão da necessidade de compreensão uniforme, por parte de vários grupos de trabalho, de diversos protocolos de comunicação de dados foi desenvolvida uma metodologia na área concretizada em linguagens formais de especificação de protocolos.

Para uniformizar os esforços neste sentido como também para encontrar a melhor forma de apresentar os padrões de protocolos, as organizações internacionais de padronização (ISO e CCITT) formaram grupos encarregados dessa tarefa específica.

A proposta deste artigo é apresentar alguns resultados dos trabalhos da ISO como também a possibilidade de aproveitamento destes resultados por grupos que já utilizam como metodolologia de trabalho outras linguagens diferentes das apresentadas pela ISO, no caso as Redes de Petri.

Os exemplos escolhidos no artigo constituem os mecanismos pertencentes ao Protocolo de Transporte Classe 2 da ISO.

2. O Enfoque da ISO

A necessidade de dispor-se de técnicas de descrição formal de serviços, protocolos e interfaces de comunicação levou a ISO a criar um grupo de trabalho com o objetivo de desenvolver técnicas de descrição formal. Este grupo se dividiu em três subgrupos (Ref [1]):

- subgrupo A: Objetiva desenvolver conceitos arquiteturais de tal forma a suportar os trabalhos dos demais subgrupos;

- subgrupo B: Objetiva desenvolver a técnica de descrição formal baseada em máquina de estados finitos estendida (Estelle);

- subgrupo C: Objetiva desenvolver a técnica de descrição formal baseada em ordem temporal de primitivas de interação.

Os conceitos desenvolvidos pelo subgrupo A são fundamentais para uma abordagem clara da especificação, independentemente da técnica utilizada.

Estes conceitos podem ser divididos em duas categorias:

1) Conceitos relacionados à arquitetura de um sistema:

- módulo
- canal
- porta

2) Divisão da especificação em:

- serviço
- protocolo
- interface

Um **módulo** representa uma unidade do sistema a ser especificado. Esta especificação pode ser feita através de alguma técnica formal ou através de um refinamento do módulo em um conjunto de submódulos interconectados, os quais por sua vez, também são objeto de especificação. Figura 1.

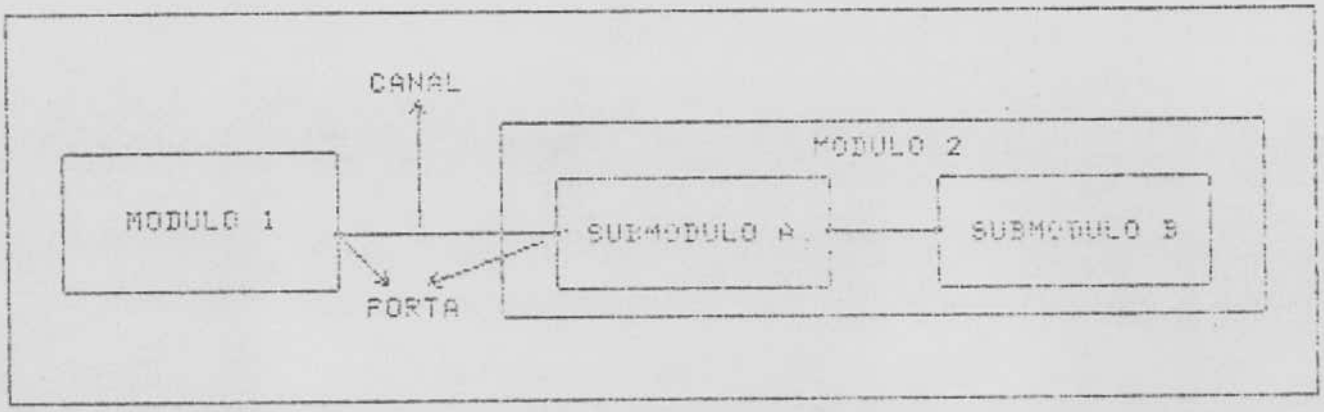

Figura 1

A interconexão de módulos se faz através dos **canais**, que se constituem do conjunto de interações que podem existir em uma ligação entre dois módulos.

Uma **porta** representa um ponto por onde um conjunto de interações definidas por um determinado canal podem ser excitadas por um módulo.

Desta forma a interconexão entre dois módulos se faz ligando uma porta de um módulo a uma porta do segundo módulo através de um canal.

A **especificação de um serviço** de comunicação se faz através da enumeração das primitivas de serviço (especificação do canal), da definição das regras locais determinando as possíveis ordem de execução das primitivas do serviço nos

pontos de acesso ao serviço (especificação das portas) e das regras globais, determinando as propriedades fim a fim do serviço de comunicação (especificação do módulo).

A especificação do protocolo pode ser considerada como um refinamento do serviço. Descreve as operações das entidades contidas no nível N em resposta aos comandos do usuário (nível N+1), as mensagens provenientes das entidades pares e as ações iniciadas internamente.

Por fim, a especificação da interface de comunicação pode ser entendida como um refinamento dos canais e portas com vistas a uma implementação. Ref [2].

3. Protocolo de Transporte Classe 2 da ISO

3.1. Características do Protocolo de Transporte

A camada de transporte é o primeiro nível a executar uma conexão fim a fim, dentro da estrutura do modelo de referência para interconexão de sistemas abertos da ISO.

De uma maneira geral, o nível de transporte oferece os seguintes serviços:

- estabelecimento de conexão de transporte;
- transferência de dados;
- liberação de conexão de transporte;
- seleção da qualidade de serviço.

Com relação às funções executadas por esta camada, citam-se:

- mapear endereços de transporte em endereços de rede;
- multiplexar as conexões de transporte em conexões de rede;
- controlar sequência e fluxo de mensagens fim a fim;
- segmentar e remontar as unidades de dados do protocolo do nível de sessão;
- concatenar e separar TPDUs (Transport Protocol Data Unist) em NSDUs (Network Service Data Units);
- fragmentar ("splitting") e recombinar TPDUs.

Porém nem todas estas funções estão disponíveis nas diferentes classes de transportes. Na classe 2 existem as seguintes características:

- multiplexação de diversas conexões de transporte em uma única conexão de rede;
- concatenação e separação de TPDUs;
- conexões de transporte com ou sem controle de fluxo explícito;

- ausência de detecção ou recuperação de erros;
- finalização de uma conexão de transporte sem o procedimento normal de liberação com a outra entidade par, na ocorrência de uma falha na conexão de rede.

3.2. Arquitetura da Especificação do Protocolo de Transporte

Considerando-se os resultados do subgrupo A da [illegible] para se iniciar uma especificação formal do Protocolo de Transporte Classe 2 deve-se definir a arquitetura da especificação.

O primeiro nível da arquitetura está representada na figura 2.

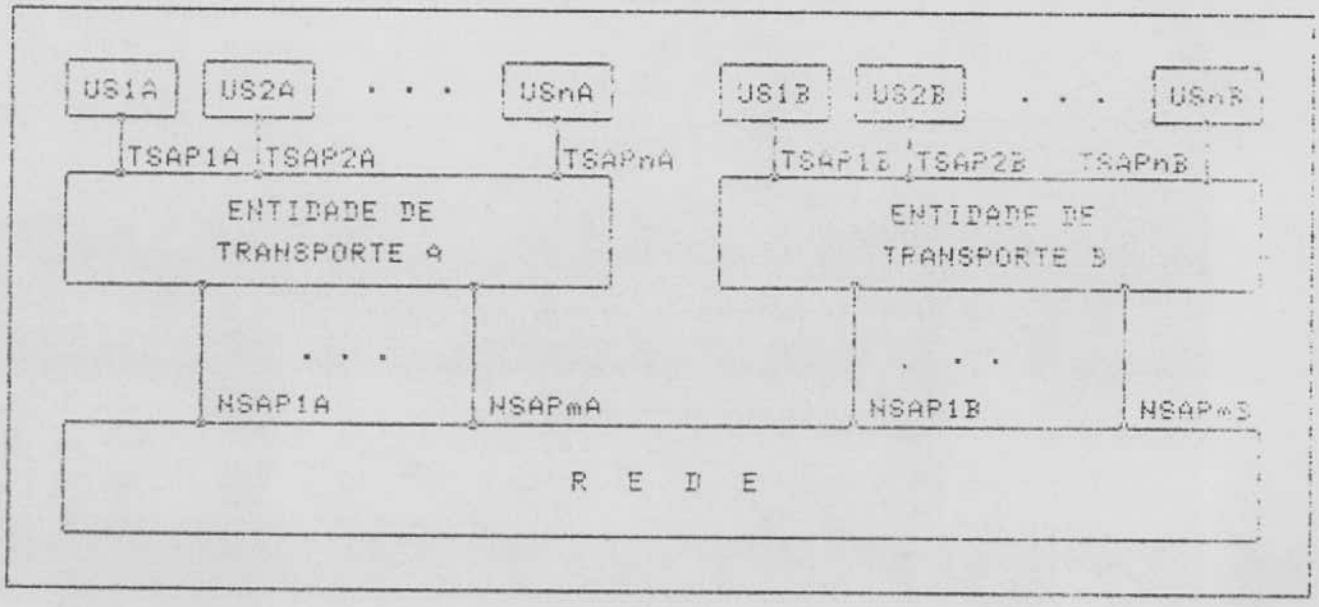

Figura 2: Arquitetura do Protocolo de Transporte.

A figura 2 representa a camada de transporte como sendo composta por diversas entidades, cada qual com o seu conjunto de usuários.

O refinamento do módulo Entidade de Transporte conduziu à representação dada pela figura 3.

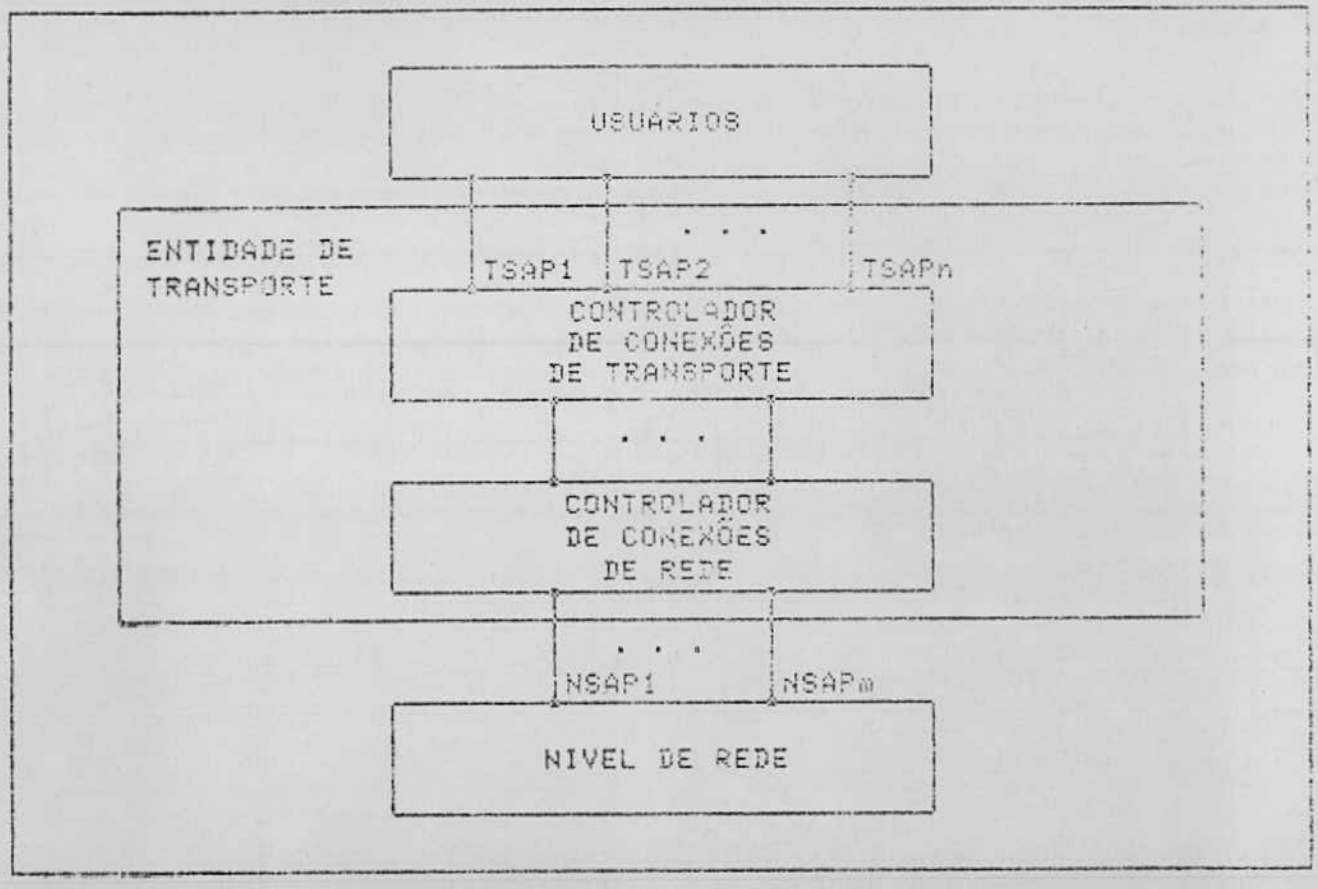

Figura 3: Refinamento da Entidade de Transporte.

Para facilidade de descrição do protocolo dividiu-se a Entidade de Transporte em dois gerenciadores: CCT (Controlador das Conexões de Transporte) e CCR (Controlador das Conexões de Rede).

Na divisão adotada neste trabalho o CCT fica responsável pelo controle das conexões de transporte e pela interface com o nível superior. A interface com o nível de rede é feita pelo CCR.

O CCR é responsável pelo controle das conexões de rede e pela interface com o NSP (Network Service Provider). A multiplexação de conexões de transporte em uma conexão de rede, a concatenação de TPDUs em NSDUs e a separação de NSDUs em TPDUs são funções do CCR. Fica portanto transparente ao CCT a multiplexação das conexões de rede, o que é muito razoável.

Essa divisão lógica da Entidade de Transporte tornou mais claro o mecanismo de multiplexação de uma conexão de rede em várias conexões de transporte, e o mecanismo de concatenação e separação.

4. Descrição do Protocolo de Transporte

A descrição do protocolo foi feita por dois grupos: o primeiro utilizou a linguagem Estelle e o segundo as Redes de Petri.

Apesar dos grupos terem trabalhado independentemente chegaram a uma mesma arquitetura, apresentada no item anterior. Entretanto, da comparação das descrições verificou-se algumas diferenças na interface entre os módulos CCT e CCR.

A descrição em Estelle apresenta um único canal entre os módulos e deixa a cargo do CCT basicamente o tratamento de conexão fim a fim.

A descrição em Redes de Petri apresenta o número de canais entre os módulos igual ao número de NSAPs. Nesta descrição o CCT foi especificado segundo as funções presentes na Ref [3], e o CCR trata das particularidades relativas à classe 2 (multiplexação e concatenação).

Neste artigo apresentaremos o CCT em linguagem Estelle e o CCR em Redes de Petri.

4.1. Descrição do Controlador de Conexões de Transporte em Estelle

Os trabalhos do subgrupo B referem-se a uma linguagem de especificação baseada em Pascal que utiliza a técnica de Máquina de Estados Finitos Estendida. Esta Técnica corresponde a uma extensão de uma máquina de estados finitos. Uma MEFE contém, além da variável de estado principal da máquina, outras variáveis, ditas de contexto, sobre as quais se pode decidir sobre transições da máquina principal.

Esta técnica permite evitar a explosão do número de estados, que frequentemente é verificada em uma MEF.

A especificação de um sistema nesta linguagem segue os seguintes passos:

SISTEMA: . Definição das CONSTANTES
. Definição dos TIPOS
. Definição dos CANAIS
. Definição dos MÓDULOS:
.. Definição das CONSTANTES
.. Definição dos TIPOS
.. Definição dos CANAIS internos
.. Definição das VARIÁVEIS
.. Definição do "STATE_SET"
.. Definição dos Procedimentos e Funções
.. Iniciações
.. Definição das Transições

A definição das constantes e tipos é análoga à linguagem Pascal.

A definição de CANAIS se faz como no exemplo abaixo:

```
channel
  TS_access_point ( TS_user, TS_provider );
    by TS_user:
      T_CON_req     ( Called_addr   : called_addr_type;
                      Calling_addr  : calling_addr_type;
                      ED_option     : ED_option_type;
                      Quality       : quality_type;
                      TS_U_data     : TS_U_data_type );

      T_CON_resp    ( Resp_addr     : Resp_addr_type;
                      Quality       : quality_type;
                      ED_option     : ED_option_type;
                      TS_U_data     : TS_U_data_type );

      T_DIS_req     ( TS_U_data     : TS_U_data_type );

      T_DATA_req    ( TS_u_data     : TS_U_data_type );

      T_EXDATA_req  ( TS_U_data     : TS_U_data_type );

  by TS_provider:
       T_CON_ind    ( Called_addr   : called_addr_type;
                      Calling_addr  : calling_addr_type;
                      ED_option     : Ed_option_type;
                      Quality       : Quality_type;
                      TS_U_data     : TS_U_data_type );

      T_CON_conf    ( Resp_addr     : Resp_Addr_type;
                      Quality       : Quality_type;
                      ED_option     : ED_option_type;
                      TS_U_data     : TS_U_data_type );

      T_DIS_ind     ( DIS_reason    : DIS_reason_type;
                      TS_U_data     : TS_U_data_type );

      T_DATA_ind    ( TS_U_data     : TS_U_data_type );

      T_EXDATA_ind  ( TS_U_data     : TS_U_data_type );
```

Esta definição indica que o canal TS_acess_point possui dez primitivas: T_CON_req, T_CON_resp, TDIS_req, T_DATA_req, T_EXDATA_req, T_CON_ind, T_CON_conf, T_DIS_ind, T_DATA_ind e T_EXDATA_ind. Destas dez primitivas, as cinco primeiras são iniciadas pelo módulo que fizer o papel de TS_user e as cinco últimas pelo módulo que fizer o papel de TS_provider.

Para cada uma das primitivas são enumerados os seus parâmetros com os respectivos tipos.

Este canal assim definido será utilizado para a conexão a um ponto de serviço (porta) do protocolo de transporte, ou seja, a conexão do usuário do serviço (TSU) ao módulo CCT.

Analogamente podem ser definidos os canais que ligam o CCT ao CCR e o CCR ao nível de rede (NSP).

A definição dos módulos se faz como no exemplo abaixo:

```
module CCT ( PORTA_CCR : CCR_access_point ( user );
             PORTA_CCT : array [1..N_acessos] of
                         TS_access_point ( TSprovider ) );
```

Esta definição indica que o módulo CCT possui uma porta PORTA_CCR através da qual o módulo se utiliza das primitivas definidas para o canal CCR_access_point e faz o papel de user. Pelas portas PORTA_CCT (existem N_acessos portas) o módulo se utiliza das primitivas definidas para o canal TS_access_point e faz o papel de TS_provider.

A partir deste ponto são declarados as variáveis do módulo, os procedimentos, as funções, etc.

Uma das variáveis do módulo (ESTADO) corresponderá ao estado principal da máquina e é sobre ela que serão especificadas as transições.

A máquina de estados do módulo CCT pode ser visualizada na figura 4.

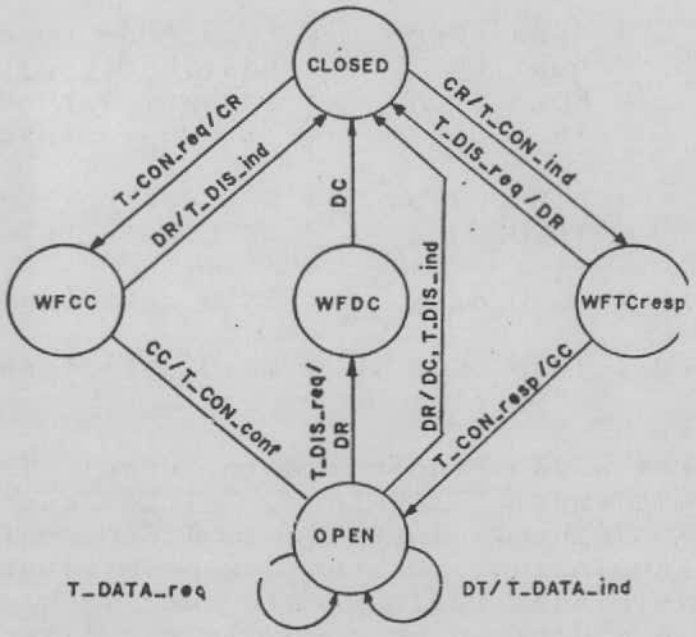

Figura 4: Máquina de estados do CCT.

Um exemplo de transição, que indica a passagem do estado CLOSED para WFCC quando ocorre a primitiva T_CON_req seria:

```
trans
  when PORTA_CCT [i] .T_CON_req
    with contexto [i] do
      from CLOSED to WFCC
        begin
          monta os parâmetros necessários à PDU CR
          out PORTA_CCR CR
        end
```

Para compreender a especificação desta transição é necessário que tenham sido feitas as seguintes declarações:

```
type
  ctx_type = record
               ESTADO : ( CLOSED, WFCC, WFTCRESP, WFDC, OPEN )
               .
               .
               .
             end;

var
  i        : 1..N_acessos;
  contexto : array [1..n_acessos] of ctx_type;
```

Com estas declarações fica definido que existe um "contexto", que se constitue da variável ESTADO e todas as demais variáveis necessárias para a especificação, para cada um dos n_acessos pontos de acesso.

Assim fica claro, na especificação da transição, que quando ocorrer a primitiva T_CON_req no ponto de acesso i, haverá uma mudança de estado aplicado à variável ESTADO dada por contexto [i].

De forma análoga devem ser especificadas todas as transições da máquina de estados do módulo.

A linguagem ESTELLE definida pelo subgrupo B contém várias outras clausulas permitidas nas transições como: provided, delay, etc., além das clausulas when, with e from utilizadas no exemplo.

Algumas considerações importantes sobre a linguagem:

- Em um dado momento várias transições podem estar habilitadas simultaneamente. Este estado de contenção pode ser desolvido na especificação através da clausula **priority** e dando uma determinada prioridade a cada transição.

- O documento da ISO cita que várias saídas (primitivas de interação geradas por uma transição) devem ser enfileiradas. No entanto, este tópico é pouco claro e não é explicado quando é feita a manipulação destas filas.

- Transições espontâneas podem ser geradas através da clausula delay, que pode criar um atraso para que uma determinada transição ocorra. Este recurso é utilizado para fins de temporização.

4.2. Descrição do Protocolo de Transporte em Redes de Petri

4.2.1. Extensões de Rede de Petri para protocolos

Rede de Petri é um modelo abstrato e formal de fluxo de informação. É um método muito poderoso para descrição e análise dos fluxos de informação e controle em sistemas, particulamente sistemas que apresentam atividades assíncronas e concorrentes. Ref [4,5].

As extensões a serem apresentadas foram utilizadas com a finalidade de: ter uma descrição hierárquica, facilitar o entendimento da lógica dos processos, destacar as trocas de mensagens entre os processos e simplificar a representação.

4.2.1.1. Transições Condicionais Simples

São transições sujeitas a um teste lógico. Dependendo do resultado do teste ser verdadeiro ou falso é colocado uma "marca" no "lugar" correspondente ao resultado da condição.

Figura 5: Exemplo de transição condicional simples.

4.2.1.2. Transições Condicionais Múltiplas

Esta extensão é uma generalização da anterior. Também aqui as transições são acompanhadas de uma condição, que poderá ter vários resultados. Para cada resultado da condição a transição apresenta uma saída para um "lugar". Durante a execução da rede de Petri apenas o "lugar" correspondente ao resultado da condição recebe a "marca".

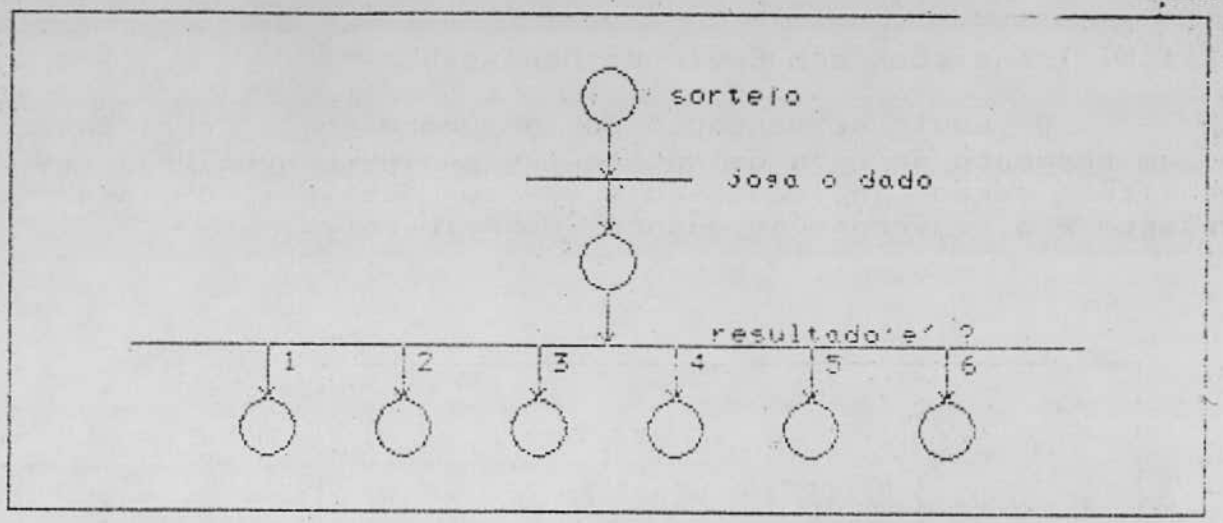

Figura 6: Exemplo de transição condicional múltipla.

4.2.1.3. Blocos Funcionais

Bloco funcional é uma rede de Petri que se encontra embutida em outra rede de Petri, permitindo a representação de uma rede, relativamente grande, em uma única página.

As regras para utilização de blocos funcionais são as seguintes:

a. Um bloco funcional começa num arco direto vindo de uma transição, e termina com um arco direto indo para outra transição. O bloco funcional corresponde pois a um "lugar" da rede que está inserido.

b. Um bloco funcional só possui uma entrada e uma ou mais saídas.

4.2.1.4. Lugares que representam ocorrência de mensagens

O "lugar" correspondente ao estado de espera de mensagem é representado por um retângulo ao invés de círculo, e nele é detalhado o processo que originou a mensagem, o tipo da mensagem e o seu conteúdo quando for relevante.

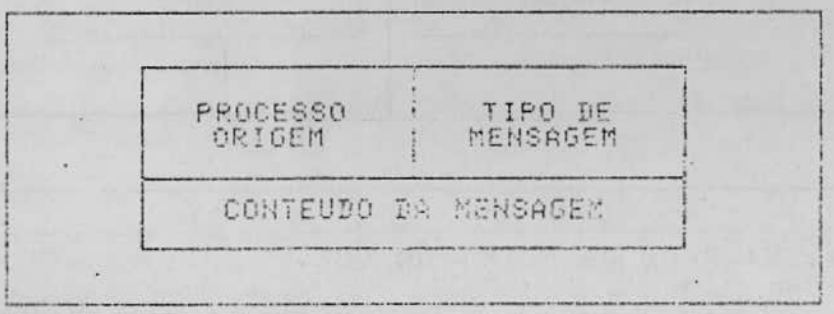

Figura 7: Formato de "lugares" de ocorrência mensagens.

4.2.1.5. Transições com Envio de Mensagens

O envio de mensagem por um processo é representado por um segmento de reta orientado (em qualquer sentido). Nesta transição serão indicados o processo destino, o tipo de mensagem e o seu conteúdo quando for relevante.

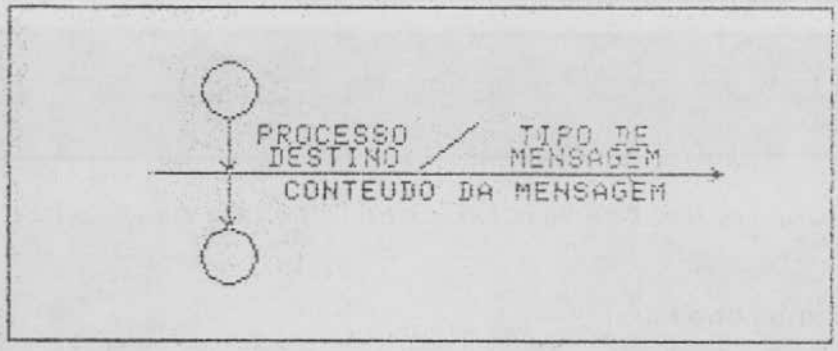

Figura 8: Transição com envio de mensagem.

4.2. Descrição do Controlador de Conexões de Rede

A rede de Petri do Controlador de Conexões de Rede tem como dois grandes blocos funcionais o CCR_NSP e o CCR_CCT.

O CCR_NSP cuida dos eventos provenientes do NSP, enquanto que o CCR_CCT cuida dos provenientes do CCT.

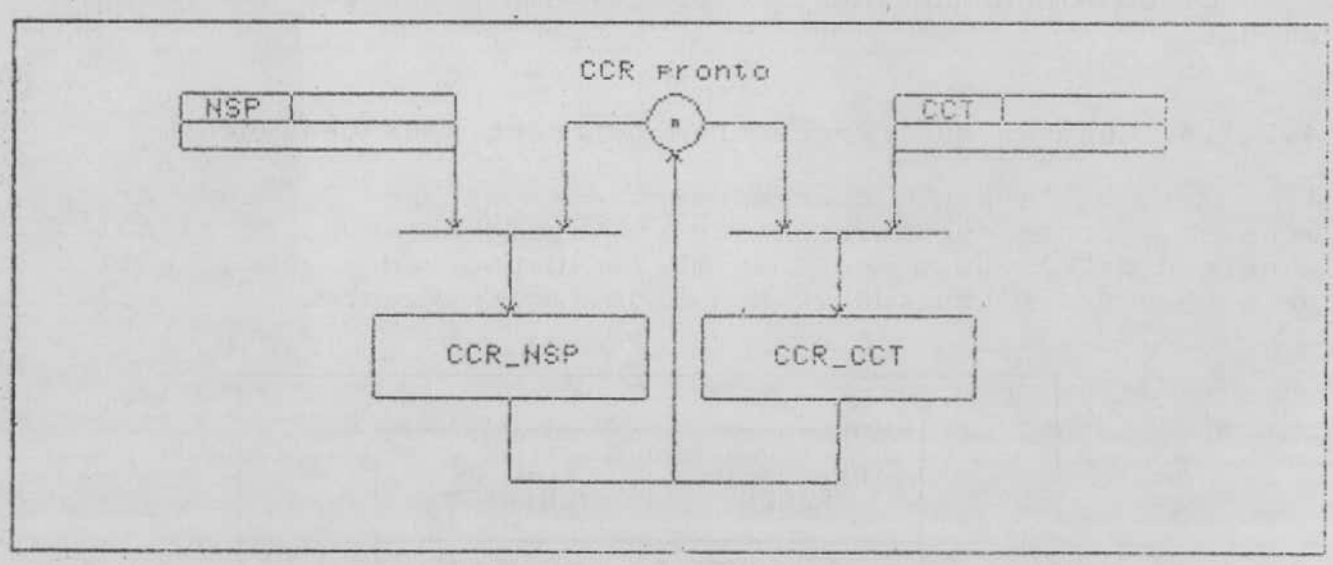

Figura 9: Rede de Petri do CCR.

4.2.1. Descrição do bloco funcional CCR_NSP

Este bloco cuida das ativações do Network Service Provider.

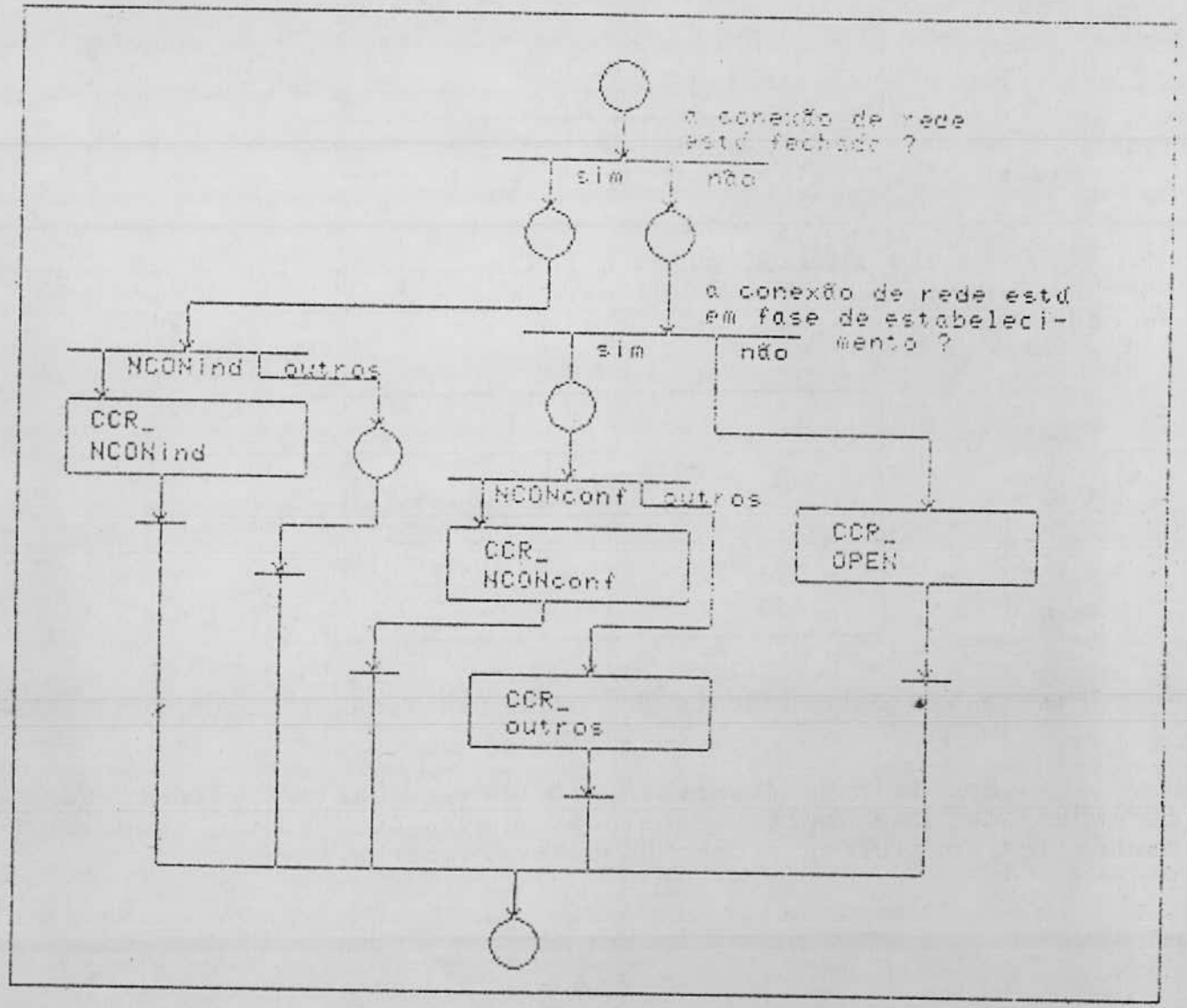

Figura 10: Detalhamento do bloco funcional CCR_NSP.

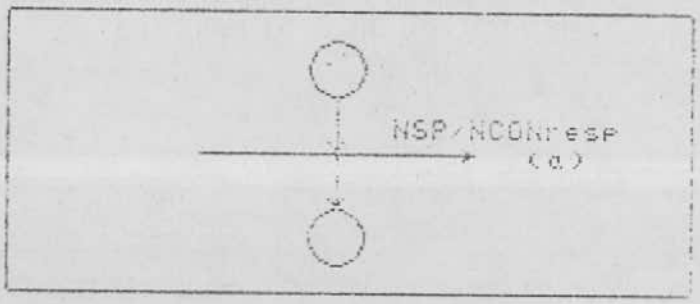

Figura 11: Detalhamento do bloco funcional CCR_NCONind.

Em (a) da figura 11, o CCR responde a solicitação de estabelecimento da conexão de rede e irá considerar a conexão de rede aberta.

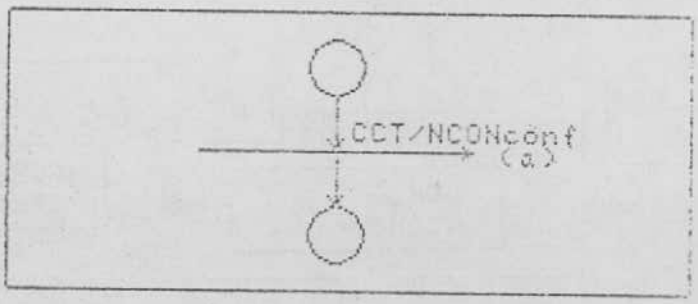

Figura 12: Detalhamento do bloco funcional CCR_NCONconf.

Em (a) da figura 12, O CCR irá ativar todas as conexões de transporte associadas a essa conexão de rede e também irá considerar a conexão de rede aberta.

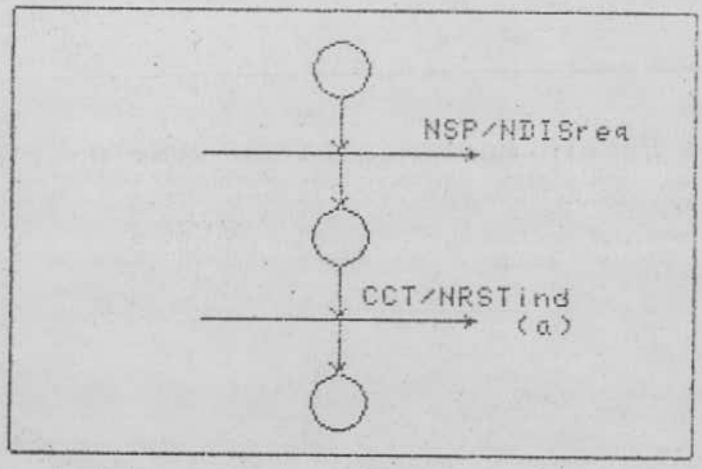

Figura 13: Detalhamento do bloco funcional CCR_outros.

Em (a) da figura 13, o CCR irá ativar todas as conexões de transporte associadas a essa conexão de rede, sinalizando a "falha" da conexão de rede e considerando fechada a conexão de rede. A ativação NFAIL foi utilizada visando caracterizar um procedimento não esperado pela entidade de transporte, resultando na liberação da conexão de rede e consequentemente de todas as conexões de transporte associadas.

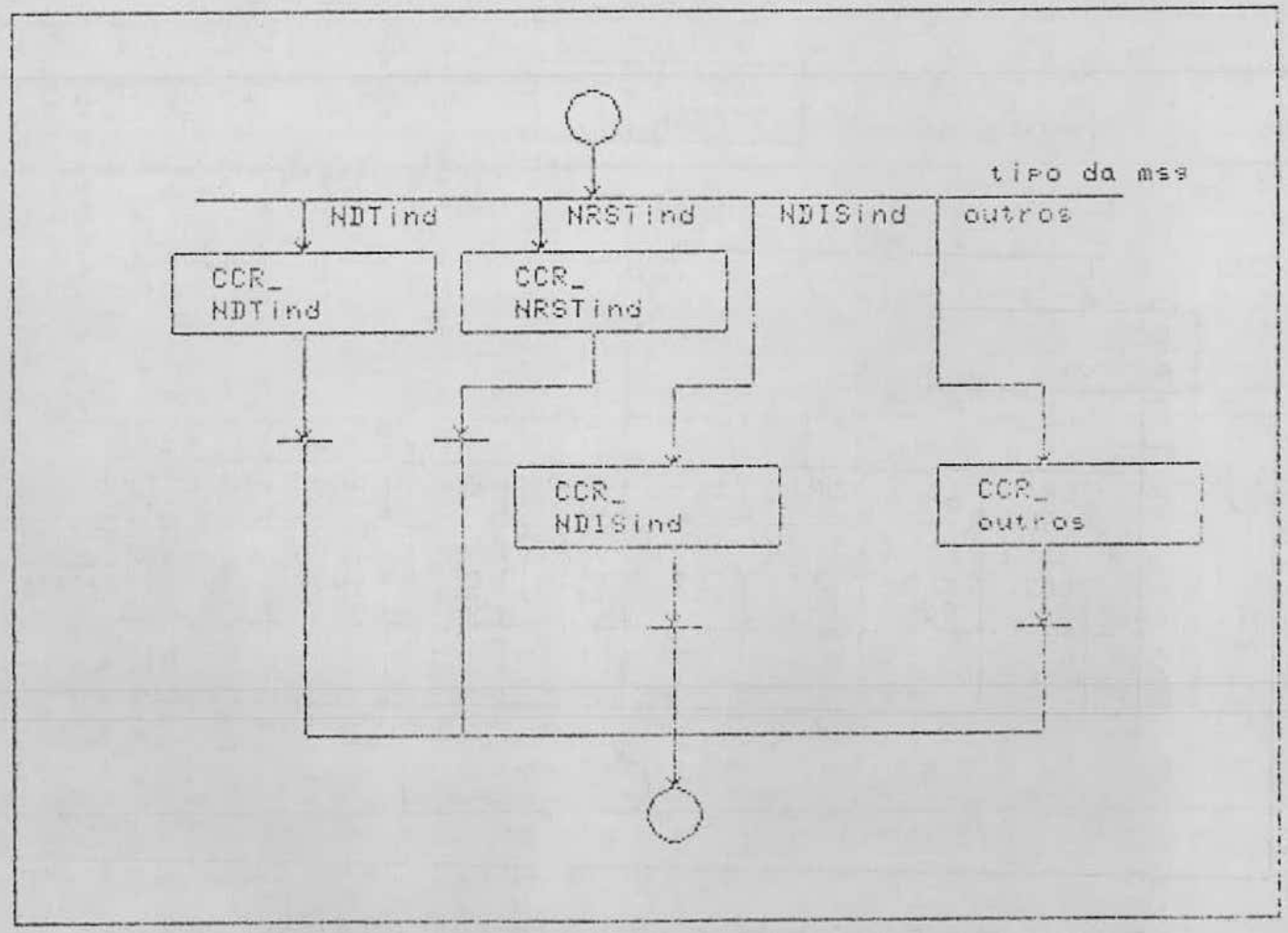

Figura 14: Detalhamento do bloco funcional CCR_OPEN.

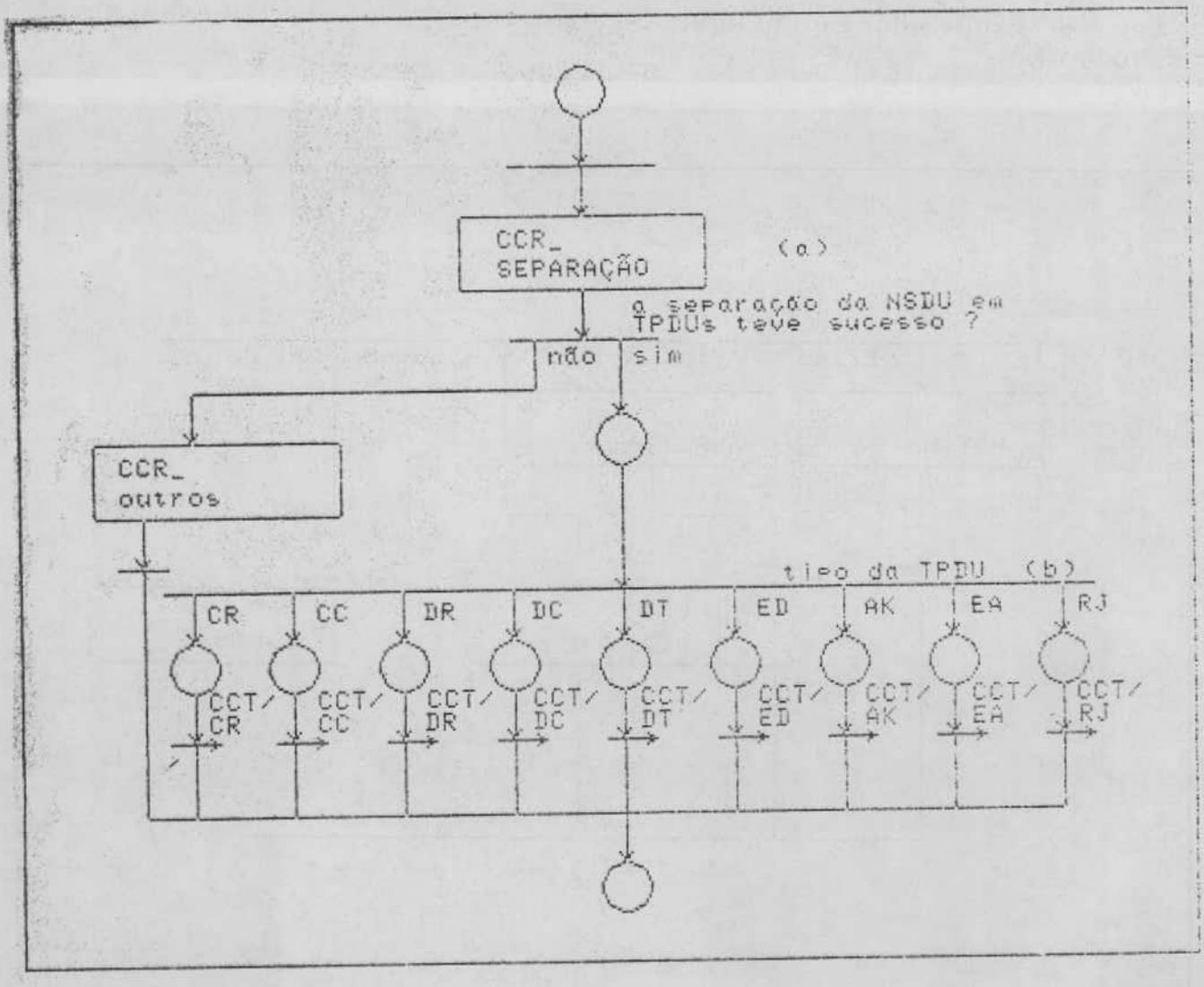

Figura 15: Detalhamento do bloco funcional CCR_NDTind.

Em (a) da figura 15, o CCR irá executar o procedimento de separação segundo as regras definidas no ítem 6.4 da Ref [3]. Em (b), o CCR irá encaminhar todas as TPDUs resultantes dessa separação, com a ativação da conexão de transporte específica.

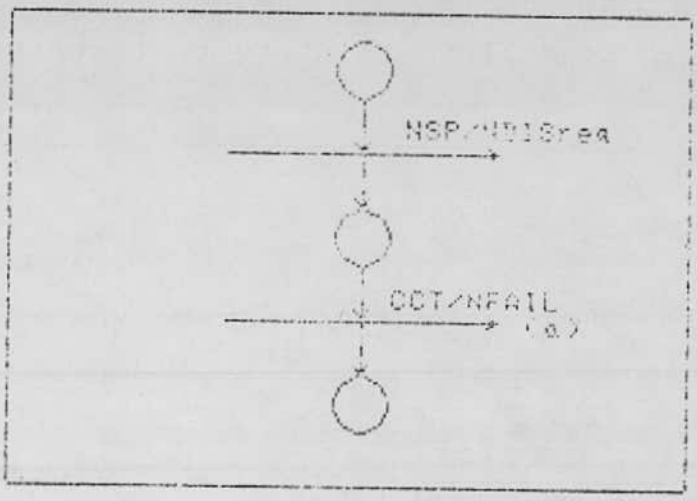

Figura 16: Detalhamento do bloco funcional CCR_NRSTind.

Em (a) da figura 16, o CCR deverá ativar todas as conexões de transporte associadas a essa conexão de rede, sinalizando o evento NRSTind.

Neste procedimento está sendo assumido de que a entidade de transporte irá liberar a conexão de rede que iniciou o reset. A opção de continuar o procedimento de reset não foi considerada.

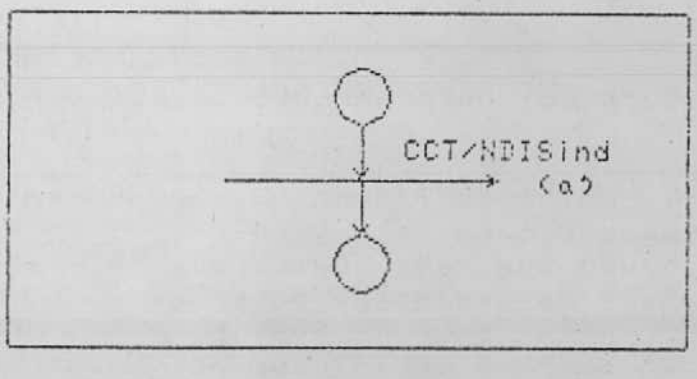

Figura 17: Detalhamento do bloco funcional CCR_NDISind.

Em (a) da figura 17, o CCR deverá ativar todas as conexões de transporte associadas a essa conexão de rede, sinalizando o evento NDISind.

4.2.2. Descrição do bloco funcional CCR_CCT

Esse bloco cuida do tratamento das ativações do CC

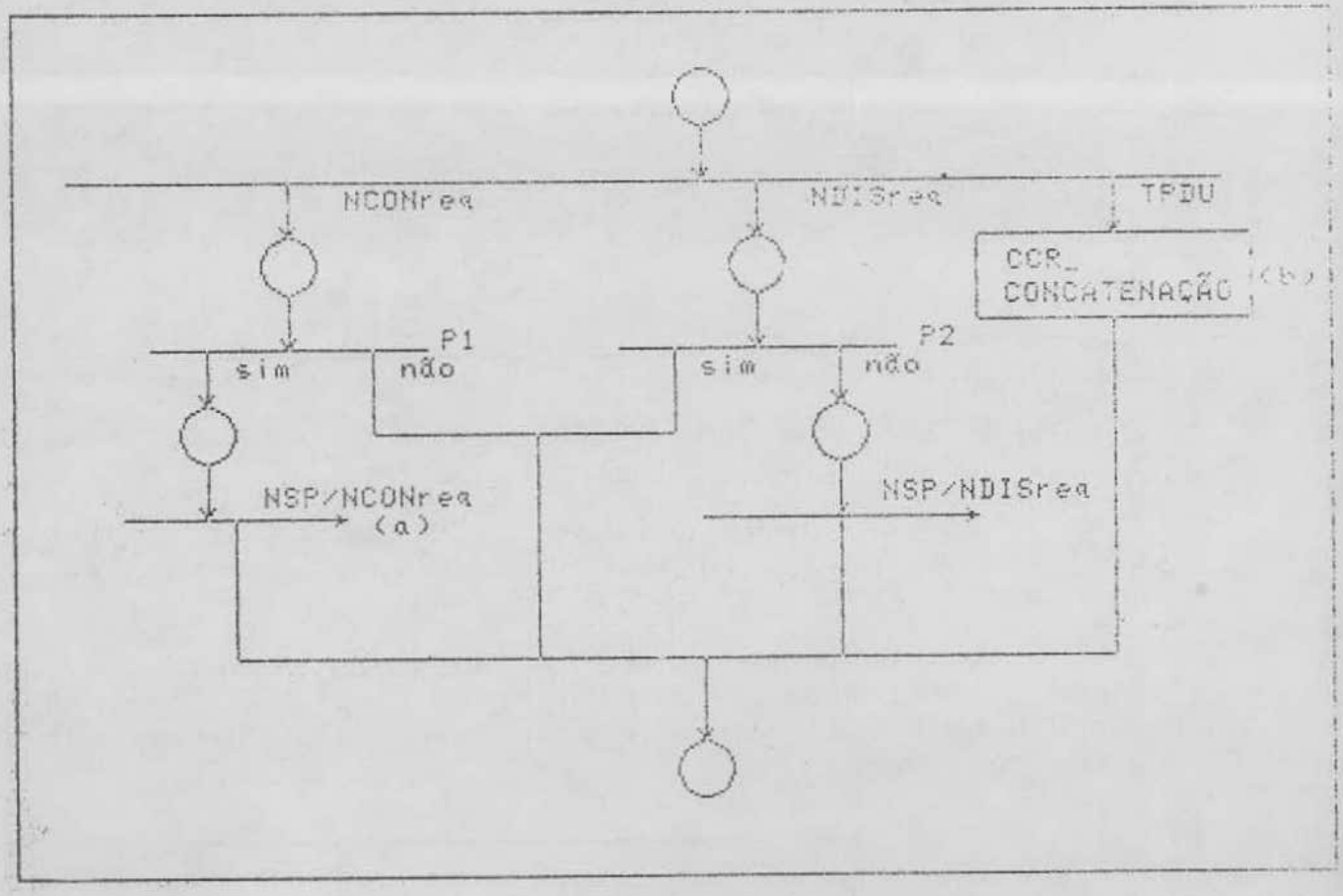

Figura 18: Detalhamento do bloco funcional CCR_CCT.

Em P1 da figura 18, o CCR verificará se a conexão de rede especificada está fechada. Isto é feito para evitar que uma conexão de rede que esteja em fase de estabelecimento receba mais um pedido de conexão.

Em P2 é verificado se há outra conexão de transporte associada a esta conexão de rede. Isto para que apenas quando a última conexão de transporte multiplexada for liberada haja a leberação da conexão de rede.

Em (a), o CCR irá considerar que a conexão de rede especificada entrou em fase de estabelecimento.

Em (b) é executado o procedimento de concatenação de TPDUs de diferentes conexões de transporte multiplexadas na mesma conexão de rede em apenas uma NSDU, segundo as regras descritas no item 6.4 da Ref [3].

5. Conclusão

A experiência adquirida através das atividades descritas permite concluir os seguintes pontos:

- As técnicas de descrição formal propostas pela ISO devem ser utilizadas na especificação de protocolos padronizados para fins de divulgação. Com essa preocupação evitam-se múltiplas interpretações do mesmo protocolo.

- Os resultados do subgrupo A podem ser aproveitados independentemente da linguagem de especificação adotada (Estelle, Lotus, Redes de Petri, etc).

- A especificação de qualquer tipo de protocolo, visando uma implementação, deve partir de uma arquitetura elaborada segundo os resultados do subgrupo A. O fato da ISO recomendar o uso das linguagens Estelle e Lotus, na descrição dos módulos definidos na arquitetura, não implica na necessidade de mudança de metodologia de trabalho dos grupos já acostumados com outras linguagens (por exemplo Redes de Petri).

- A especificação formal de um mesmo protocolo padronizado, tendo como modelo de referência uma especificação segundo as recomendações da ISO, feita por dois grupos: um utilizando a linguagem Estelle e outro Redes de Petri pode conduzir a implementações diferentes em vários equipamentos, mas permitem um funcionamento integrado num ambiente de sistemas distribuido.

Referências

1. Chris A. Vissers, Richard l. Tenney, Gregor v. Bochmann
Formal Description Techniques
Proceedings of the IEEE, vol. 71, No. 12, December 1983.

2. Wanderley Lopes de Souza
Utilização dos conceitos de módulo, porta e canal em especificações formais de serviços, protocolos e interfaces de comunicação.
3. Simpósio Brasileiro sobre Redes de Computadores
Rio de Janeiro - abril de 1985

3. Draft Proposal ISO / DP 8073
Information Processing System - Open Systems Interconnection - Connection Oriented Transport Protocol Specification

4. James L. Peterson. Petri Nets.
Computing Surveys, vol. 9, No. 3, September 1977.

5. Estrutura do Software do PAD. P.119.RT.013.1.
FDTE - Telebrás - 01/03/1982.