ESPECIFICANDO PROTOCOLOS ATRAVÉS DE UMA METODOLOGIA BASEADA EM REDES DE PETRI*

Jaelson Freire Brelaz de Castro

Paulo Roberto Freire Cunha

Grupo de Redes de Computadores e
Sistemas Distribuídos (REDIS)
Departamento de Informática da UFPE
Cidade Universitária
RECIFE - PE

RESUMO

O presente trabalho propõe uma metodologia para a especificação de arquitetura hierárquicas de protocolos de comunicação entre computadores. Este objetivo é alcançado através do uso das Redes de Petri Estendidas, que se mostraram bastante úteis para o desenvolvimento e especificação destes protocolos.

ABSTRACT

This work proposes a methodology for specification of hierarchical architectures of computer communications protocols. Its goal is allowed throughout the use of Extended Petri Nets, wich seemed to be useful for the design and specification of these protocols.

(*) Este trabalho contou com auxílio parcial do CNPq.

1. INTRODUÇÃO

Entre as várias técnicas de descrição existentes escolhemos as Redes de Petri para auxiliar a tarefa de especificação dos protocolos ([Peters 81], [Berter 83]). Entre os motivos desta decisão estão: a sua simplicidade conceitual e seu poder de expressão e análise. De fato, este modelo abstrato e formal se mostra extremamente útil como ferramenta de descrição e análise do fluxo de informações e controle de protocolos, que como sabemos se caracterizam por apresentar atividades assíncronas e concorrentes.

Neste trabalho procuramos identificar uma metodologia para especificação de protocolos de comunicação, que se baseia no uso de extensões das Redes de Petri. Assumimos que a arquitetura é composta de camadas ou módulos, que interagem entre si. Sempre que se fizer necessário, a especificação de um módulo será refinada em submódulos. Estabeleceremos uma estratégia de transparência quanto ao processo de interação entre módulos adjacentes, através do uso de "Interfaces". O acesso as "Interfaces" será obtido por intermédio do "Gerenciador de Interfaces". Decidimos pela estrutura de dados FILA para representação das interfaces. Na especificação dos módulos adotaremos o modelo de programação orientado para o elemento básico do ambiente de protocolos, o evento. Sabemos que a evolução de um protocolo é caracterizada por fases bem distintas. Na nossa proposta cada uma destas fases, corresponderá a um "Estado". Associado a ocorrência dos eventos estarão ações a serem executadas de acordo com o estado do protocolo.

Finalmente, descreveremos sucintamente, a título de exemplo de utilização da metodologia proposta, a especificação do Protocolo de Acesso ao Meio DIMAC, desenvolvido para a rede local do Departamento de Informática da UFPE. A especificação completa deste protocolo pode ser encontrada em [Castro 86].

2. METODOLOGIA

2.1. Modularização

Desde as primeiras experiências de projetos para estabelecimento de um processo de comunicação entre computadores, procurou-se estruturar a tarefa de comunicação em níveis hierárquicos ou camadas ([Boch 80a], [Boch 80b], [QueCun 84]). A cada camada caberá um problema menos complexo de comunicação entre computadores. Assim uma camada, em um alto nível de abstração, poderá ser considerado um módulo do sistema que está encarregado de executar um determinado tipo de procedimento.

Considerando o ambiente das Redes de Petri, observamos que este princípio também está presente, pois muitas vezes um lugar muitas vezes representará uma abstração de um procedimento. Nesta ótica uma camada N (módulo) será modelada por um lugar especial: Módulo-N (Figura 1).

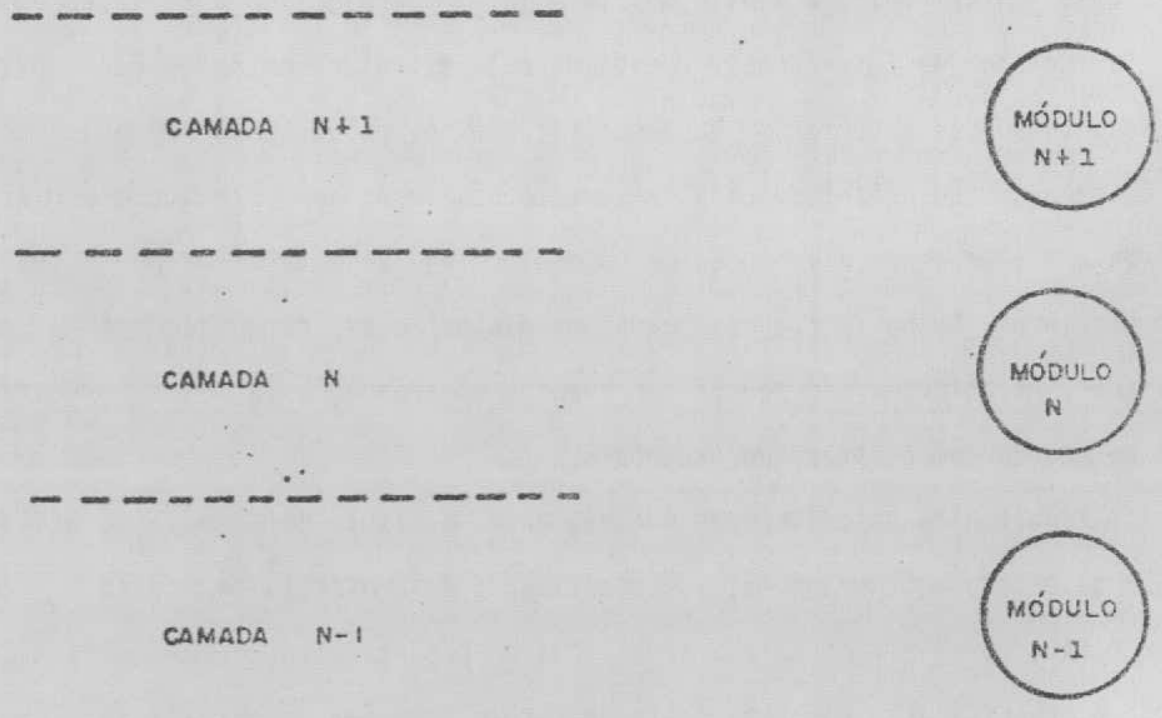

Figura 1 - Estratificação Através de Lugares

Na solução de suas tarefas complexas, os módulos necessitam interagir com os demais. A comunicação entre eles será obtida através do uso de interfaces, que tornam transparente toda as atividades de comunicação envolvidas. Note que também nos modelos de Petri esta característica se encontra presente, pois a interação entre os lugares (módulos) será realizada através do disparo de transições especiais, aqui chamadas de interfaces, representadas por barras cheias nos diagramas. A transição funcionará como "Canal de Comunicação" entre os diversos lugares (módulos) da rede (sistema).

Uma das características presentes no modelo proposto, será a capacidade de se refinar um módulo em sucessivos submódulos. Esta capacidade de refinamento pode ser necessária, quando da especificação de protocolos bastantes complexos, que necessitam portanto de refinamentos gradativos para sua melhor compreensão. Considere o exemplo genérico de um protocolo de comunicação de uma rede de computadores. Numa abstração inicial, podemos considerar uma arquitetura composta de duas camadas ou módulos; Módulos Usuários do serviço de protocolo de comunicação e o próprio Módulo Protocolo de Comunicação. A interação entre os módulos será realizada através de uma interface apropriada (Figura 2).

Quando nos referimos ao ambiente de redes de computadores, onde os usuárias estão quase sempre separados fisicamente, sabemos que existirão entidades definidas localmente a cada usuário. Uma entidade se comunicará com as entidades equivalentes (peer-entity) através dos serviços oferecidos pelos níveis inferiores. Assim o Módulo Protocolo poderá ser refinado naturalmente em dois novos submódulos: Submódulo Entidades e Submódulo Níveis Inferiores (vide Figura 3). Observe que a técnica de refinamento passo a passo poderá ser usada recursivamente. Um submódulo poderá, se necessário, derivar novos grafos que garantam o detalhamento necessário.

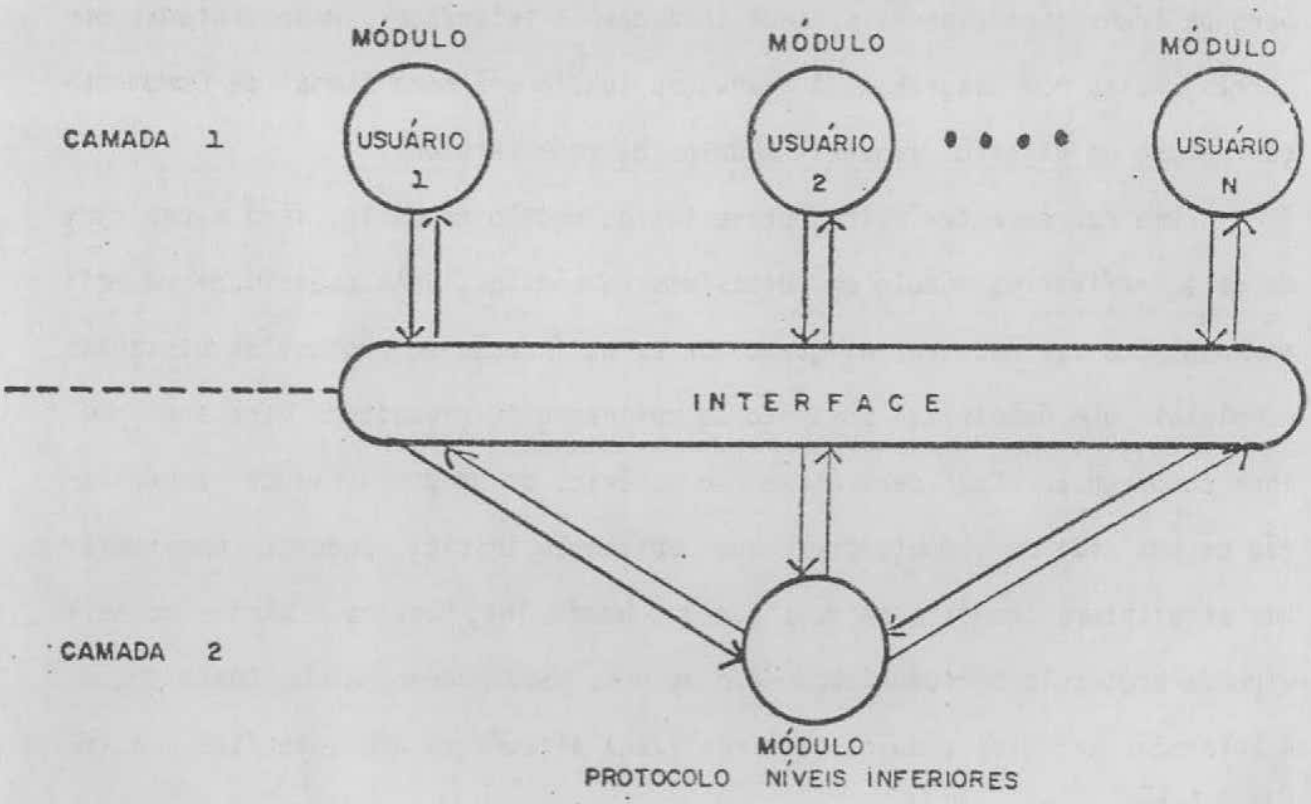

Figura 2 - Modularização de um Protocolo

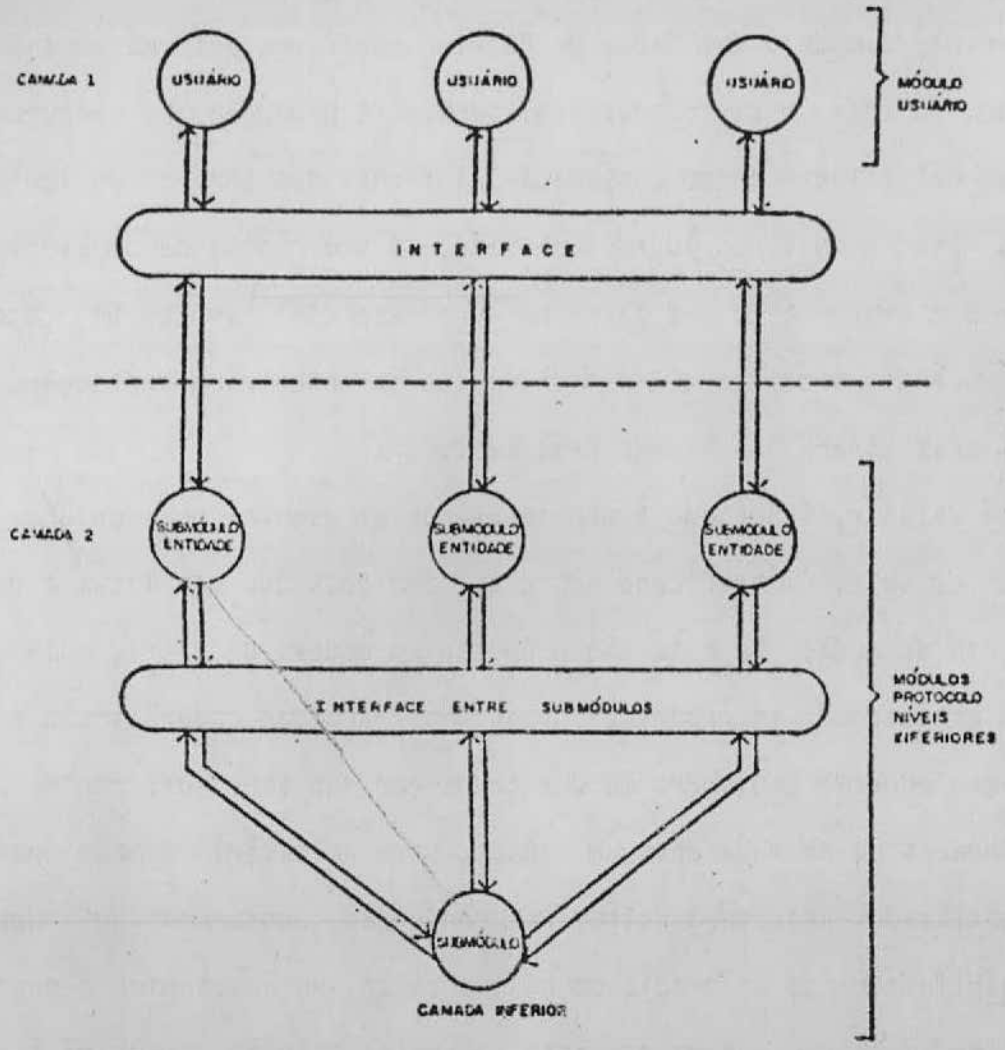

Figura 3 - Refinamento Módulo Protocolo

2.2. Especificação do Módulo

Adotaremos o modelo de programação orientado para o elemento básico do ambiente de protocolos, o evento ([Queiroz 84], [QueCun 84]). A ocorrência de um evento está associada a chegada de mensagens. Na realidade três são as "fontes" de origem das mensagens: a própria camada, a camada inferior e a camada superior. A cada evento ocorrido estarão associados conjuntos de ações que deverão ser executados. A sua execução, invariavelmente "consumirá" as mensagens que provocaram o evento e possivelmente "produzirá" novas mensa-

gens. Em síntese podemos identificar que o elemento fundamental do sistema que pretendemos especificar será o par: evento mais conjunto de ações.

Observando o modelo das Redes de Petri encontramos este mesmo tipo de comportamento, ou seja, o ciclo: disparo, consumo e produção. Na execução de uma transição habilitada, serão consumidas as fichas dos lugares de entrada e produzidas novas fichas nos lugares de saída. A ocorrência de um evento estará associada a presença de uma ficha no lugar especial Evento. De acordo com o tipo de evento ocorrido e o estado em que se encontra o protocolo, um conjunto de ações estará habilitado para execução.

Poderá existir, associado a ocorrência de um evento, uma sentença de precondição, ou seja, um predicado sobre as condições que habilitam a execução do conjunto de ações. Na extensão proposta ao modelo de Petri, esta característica está também representada, pois uma transição poderá incluir um predicado. Para ocorrer o disparo de uma transição não será suficiente apenas que os lugares de entrada possuam fichas, será necessário também que o predicado habilitador seja verdadeiro. Por convenção, adotaremos que quando não houver explicitamente um predicado na transição, será assumido o predicado "Sempre "Verdade". O diagrama proposto com esta inclusão, assumirá o seguinte aspecto (Figura 4):

Os conjuntos de ações relacionados com a ocorrência dos eventos deverão ser considerados indivisíveis ou atômicos. Assim apenas um conjunto de ações poderá ser executado por vez. Esta propriedade será respeitada através da definição de um conjunto de variáveis especiais (lugares) chamado "Estados", que como o próprio nome diz, corresponde a idéia de estado, existente num modelo de máquinas de estados finitos. Qualquer conjunto de ações para ser executado, deverá estar associado a ocorrência não apenas do evento a qual ela está relacionada, mas também a um determinado estado do protocolo. Portanto a evolução do protocolo pode ser acompanhada através da observação da sequência de estados assumidos. Ao término da execução das ações corres-

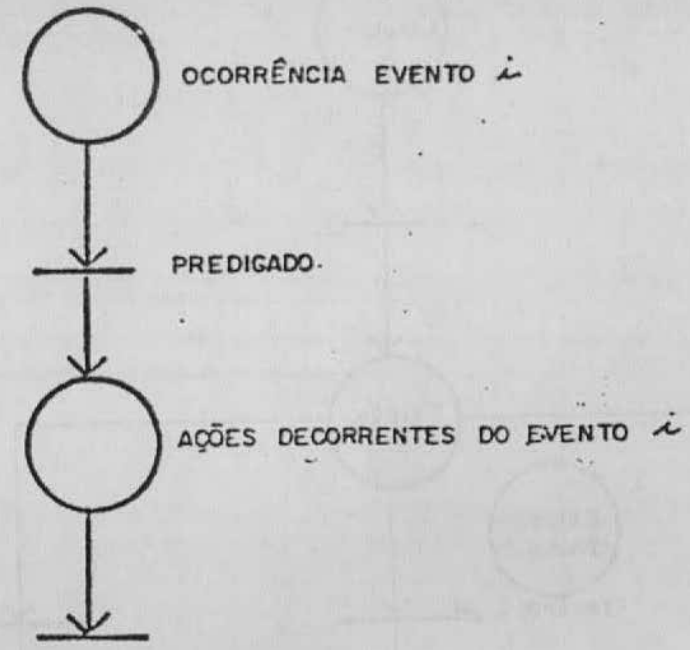

Figura 4 - Dependência do Evento

pondentes a ocorrência do evento um novo estado, Estado Final, poderá ser alcançado. O Estado Final alcançado poderá ser o mesmo do incial.

Muitas são as vantagens obtidas do uso do modelo que hora propomos. Adicionada a grande facilidade e naturalidade das descrições, temos a capacidade de estruturação da especificação. Por exemplo, se depararmos com uma situação onde para um mesmo evento ocorrido, dependendo do estado inicial, diversas ações poderão ser executadas, temos o grafo (Figura 5) simplificado.

Outra característiva fundamental na especificação de protocolos será o não determinismo na escolha do conjunto de ações a ser executado. Se mais de um conjunto de ações estiver habilitado, será realizada uma escolha não determinística de qual será executado. Note que esta característica também está presente no modelo de Petri, haja vista que por definição, quando várias transições se encontrarem habilitadas, apenas uma, escolhida aleatoriamente, será disparada.

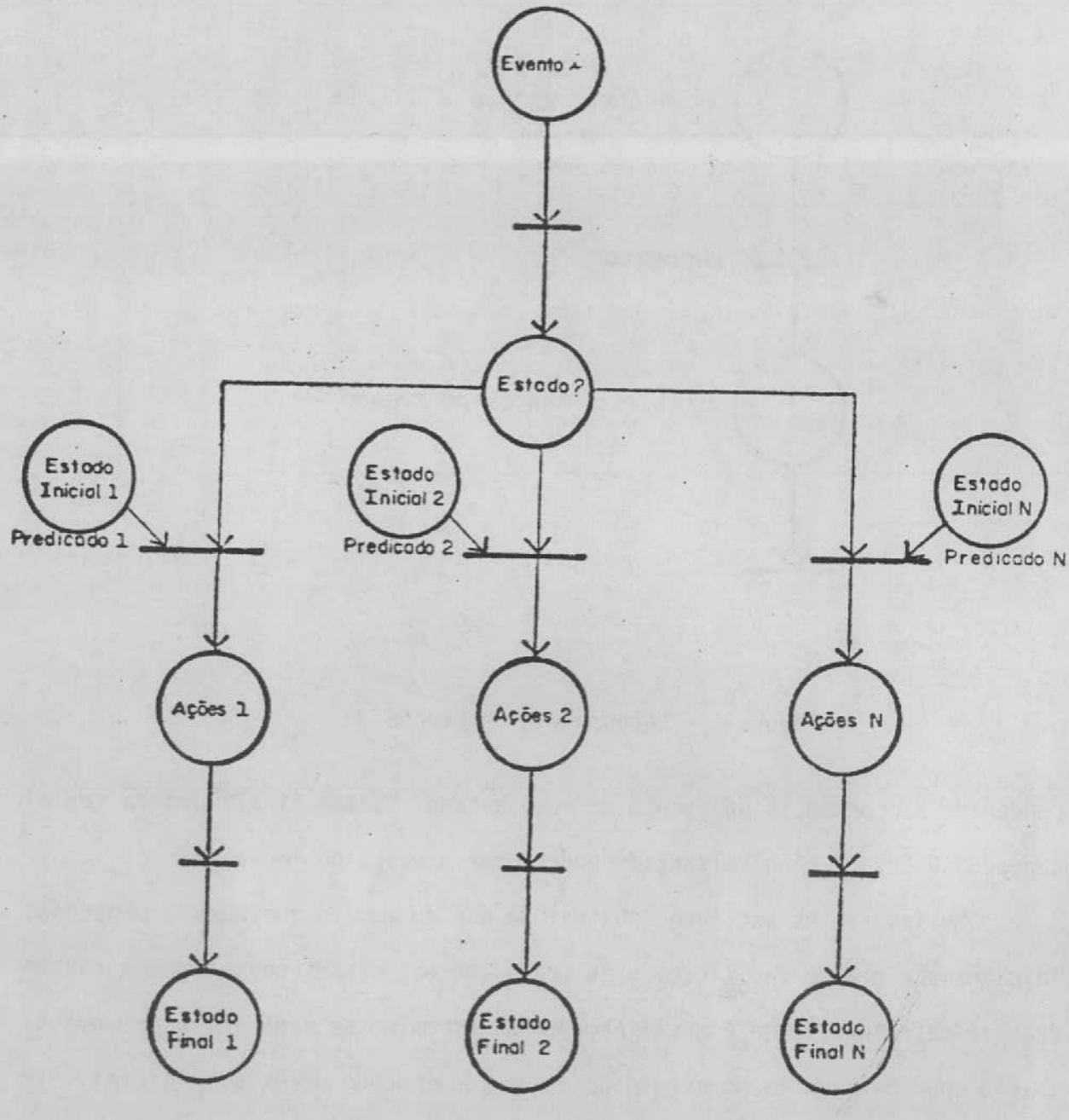

Figura 5 - Estruturação de Acordo com o Estado Inicial

Quando especificamos protocolos também se faz necessário, algumas vezes, a introdução de prioridades, ou seja estabelecer em um conjunto de eventos habilitados, qual será o mais prioritário. Assim se dois ou mais eventos estiverem habilitados simultaneamente, aquele com maior prioridade será con

siderado. Ocorrendo uma situação, pouco provável, de vários eventos com mesma prioridade estarem habilitados, será feita então uma escolha aleatória de qual será considerado. A incoorporação de prioridades ao modelo de Petri será bastante simples, através de parâmetros temporais (Figura 6). Quanto menor o tempo que uma transição exigir de suas condições de habilitação maior será a sua prioridade em relação as demais. O Evento 1 possuirá maior prioridade do que o Evento 2, pois suas condições habilitadoras de disparo precisam permanecer válidas por apenas uma unidade de tempo, enquanto que o Evento 2 precisa de duas unidades de tempo.

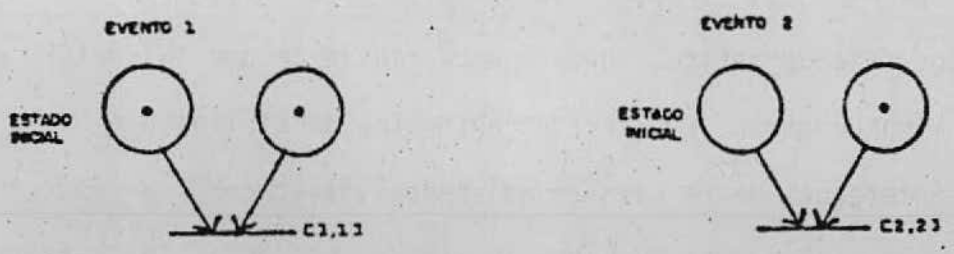

Figura 6 - Prioridades

Sabemos que em muitas situações um protocolo precisa prover a possibilidade de ações espontâneas, ou seja, aquelas que não dependem da ocorrência de eventos externos. Normalmente elas estarão relacionadas a cláusulas temporais que incluem dois parâmetros, d1 e d2. As ações só podem ser executadas após suas cláusulas de habilitação permanecerem válidas no mínimo por um tempo d1. Se continuarem verdadeiras até o tempo d2, as ações deverão ser executadas imediatamente. Note que visando não sobrecarregar os diagramas, quando a cláusula temporal não estiver presente, será assumida uma condição onde

d1=0 e d2=00 (infinito). Consequentemente as ações poderão ser executadas a qualquer instante após a sua habilitação. Denominaremos de "SemRetardo" um conjunto de ações que devem ser executadas imediatamente, ou seja os parâmetros d1=d2=0.

No modelo das Redes de Petri adotado, os parâmetros temporais de encontram por definição associados as transições. Uma transição para disparar, precisará que suas condições habilitadoras permanecam válidas por um tempo mínimo de "tmin". Caso continuem válidas até o tempo máximo permitido, "tmax", deverão ser disparadas imediatamente.

2.3. Comunicação entre Camadas

O modelo de programação por nós adotado, que se baseia no fluxo de dados, exige que o carater assíncrono e sequencial das comunicações entre as camadas ou módulos seja garantido. Surge a necessidade de uma INTERFACE entre as camadas. Visando garantir o carater abstrato, disciplinado e bastante natural para as interações entre camadas adjacentes, adotaremos a estrutura de dados FILA como elemento interfaceador. Assim se uma camada N precisa dos serviços de sua camada inferior N-1, será produzida uma mensagem naquela fila que se origina na camada N e se destina a camada inferior. A presença desta primitiva na camada inferior consumirá os parâmetros necessários e possivelmente produzirá uma nova mensagem. Uma fila possui dois pontos de acesso, um por onde são inseridos novos elementos e outro por onde são retirados. O protocolo de uma camada "observa" duas filas de primitivas de serviço em cada interface como representado na Figura 7.

Devido ao carater de independência entre as camadas, bem como a situação de interação e compartilhamento que se encontram as filas de primitivas de serviço, será necessário a introdução de um mecanismo de sincronização. Atribuiremos a este mecanismo o nome de "Gerenciador de Interface". Para o

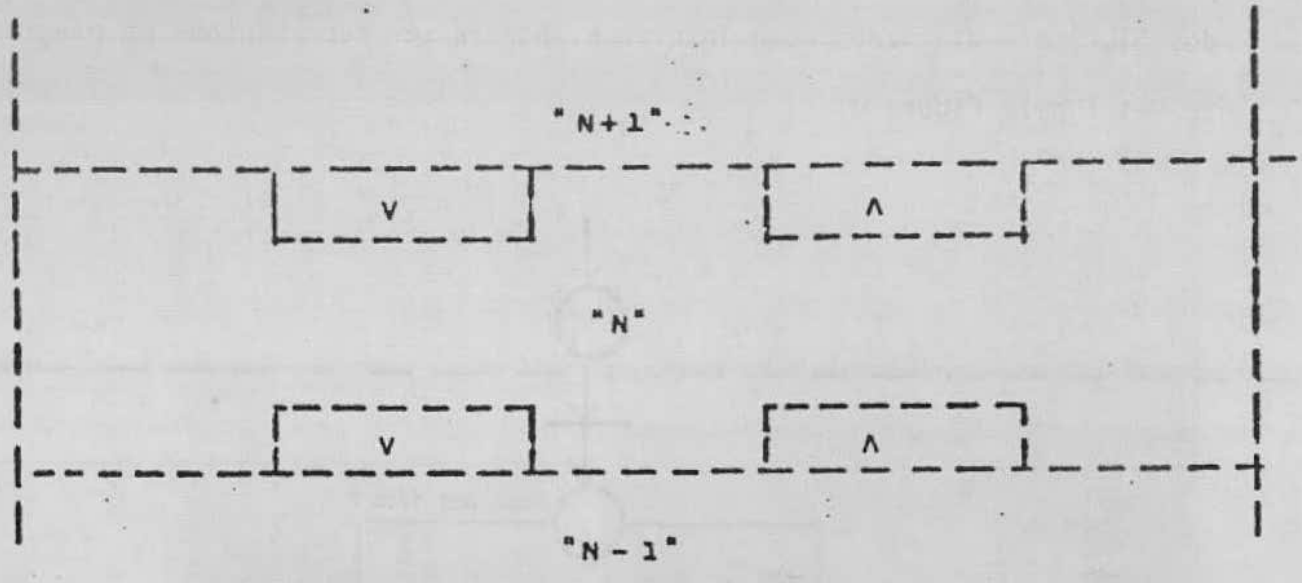

Figura 7 - Interface de uma Camada

gerenciador existirão duas filas compartilhadas: uma de transmissão e outra de recepção. O objetivo do gerenciador será disciplinar o acesso a essas filas compartilhadas de uma forma o mais transparente possível ([Queiroz 84], [QueCun 84]).

O gerenciador de interface atua sobre as duas filas presentes na interface, a de entrada e de saída, coordenando todas as interações sobre elas. As operações básicas sobre as filas são invocadas ao gerenciador correspondente, que identificará a qual das filas se destina aquela operação e como resultado produzirá uma proposta apropriada.

Apos ser identificado a qual das filas, a de entrada ou saída, está associada a operação, o gerenciador verifica qual o tipo de operação. Tratando-se de uma operação com a fila de entrada, poderá ser uma operação de remoção do primeiro elemento ou de teste se vazia. No caso de uma operação com a fila de saída será verificado se a operação é de inserção no final da fila, ou o caso excepcional de inserção no início. Ao término da operação, o gerenciador produzirá uma resposta apropriada: caso operação de teste será informado se há elemento na fila, se operação de remoção do primeiro elemento retorna este elemento e caso inserção será informado o sucesso da opera-

ração. O diagrama do Gerenciador das interfaces, no qual a estrutura de dados FILA será utilizada como interface, poderá ser representado em diagramas de Petri pela Figura 8:

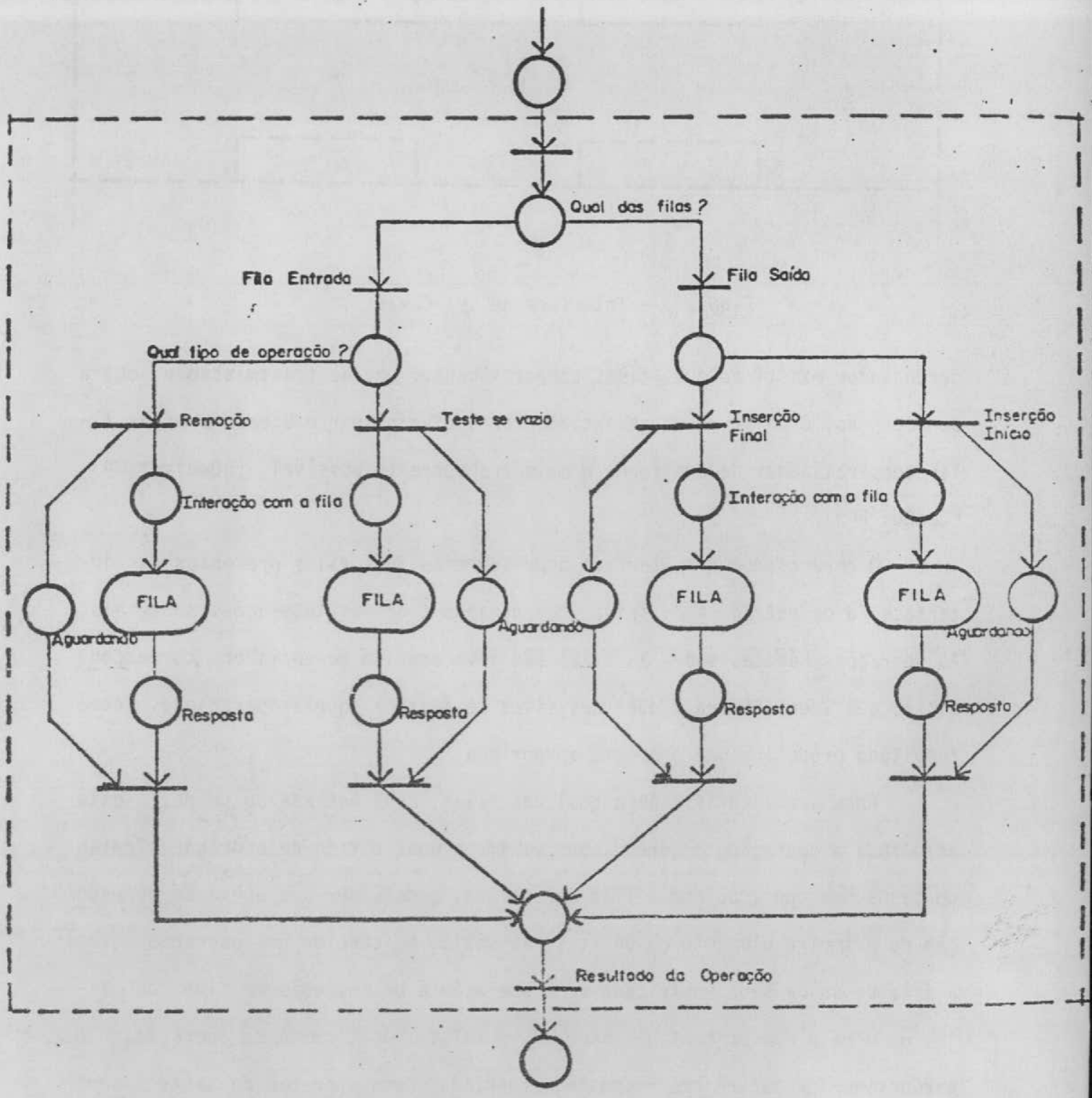

Figura 8 - Diagrama do Gerenciador

3. EXEMPLO: PROTOCOLO DE ACESSO AO MEIO DIMAC

3.1. Introdução

Entre as diversas topologias e métodos de acesso ao meio de comunicação propostos para as redes locais, o Departamento de Informática da UFPE optou por uma rede em barra com protocolo de acesso por passagem de token. No Protocolo de Acesso ao Meio DIMAC [FeCaCu 86] as estações ativas na barra formam um anel lógico, ou seja, uma sequência ordenada de estações com a última seguida da primeira. Cada estação conhece o endereço de sua sucessora e predecessora. Uma mensagem de controle, chamada token, é utilizada para controle de acesso ao meio. Esta mensagem única é repassada continuamente de estação em estação na barra. Ao concluir sua transmissão, a estação que detinha o token repassa-o para a sua sucessora. Portanto em regime permanente, a operação do DIMAC consiste na alternância de procedimentos de transferência de dados e de repasse do token.

Além do funcionamento em regime permanente, o DIMAC cuida da inicialização do anel e da gerência de situações transitórias. A inicialização se faz necessária quando as estações são primeiramente ligadas. Já as situações transitórias são causadas ou devido a entrada de novas estações no anel (aquelas que foram ligadas após a sua formação) ou devido a saída de estações (por exemplo, o usuário deseja desligar sua estação) ou mesmo decorrentes de falhas na rede. O anel se tornaria inoperante caso não fossem tomadas estas medidas de recuperação.

Sabemos que as mensagens estão relacionadas com funções bem específicas do protocolo. Procuramos a seguir identificar cada mensagem com sua função:

TOKEN - utilizada no repasse do controle de acesso a barra para sua sucessora;

QUEM-SEGUE - usada para identificar a estação que se sucede a uma determinada estação;

SOLICITA-SUCESSOR - permite as estações que possuem o endereço dentro dos limites permitidos, disputar a posse do token. Através deste mensagem novas estações podem se inserir no anel ou ser obtida a sua inicialização;

NOVO-SUCESSOR - utilizada para informar o novo sucessor da estação. Antes de se retirar do anel (deleção) a estação detentora do token envia mensagem para sua predecessora. Também durante o período de adição de novas estações ao anel, aquela que se insere, a envia para estação que iniciou a fase de adição. Uma outra possibilidade ocorre durante a fase de repasse do token, quando será possível recebê-la em resposta ao envio da mensagem QUEM-SEGUE;

DISPUTA-TOKEN - utilizada durante a fase de disputa pela posse do token;

DADOS - o campo de dados presente nesta mensagem tem origem na camada superior (LLC). Quando a camada de controle de acesso ao meio a recebe, será repassada imediatamente para a camada superior.

Buscando-se um entendimento da estrutura global do funcionamento do DIMAC, o protocolo será apresentado inicialmente através de um diagrama em blocos, que expressa as diversas fases presentes (Figura 9). Observe que por simplicidade estão omitidas as condições de transição entre os blocos.

Podemos identificar no funcionamento do protocolo de acesso duas fases principais: escuta e detenção do token. Ao serem ligadas as estações atingem a fase ou estado de escuta. Enquanto neste estado o protocolo pode evoluir para as seguintes fases:

- Recepção, caso endereçado por uma mensagem DADOS, oriunda da camada de controle de enlace lógico, retornando em seguida para a fase de escuta;
- Detenção do Token, caso endereçado por uma mensagem de TOKEN ou por uma mensagem QUEM-SEGUE válida.

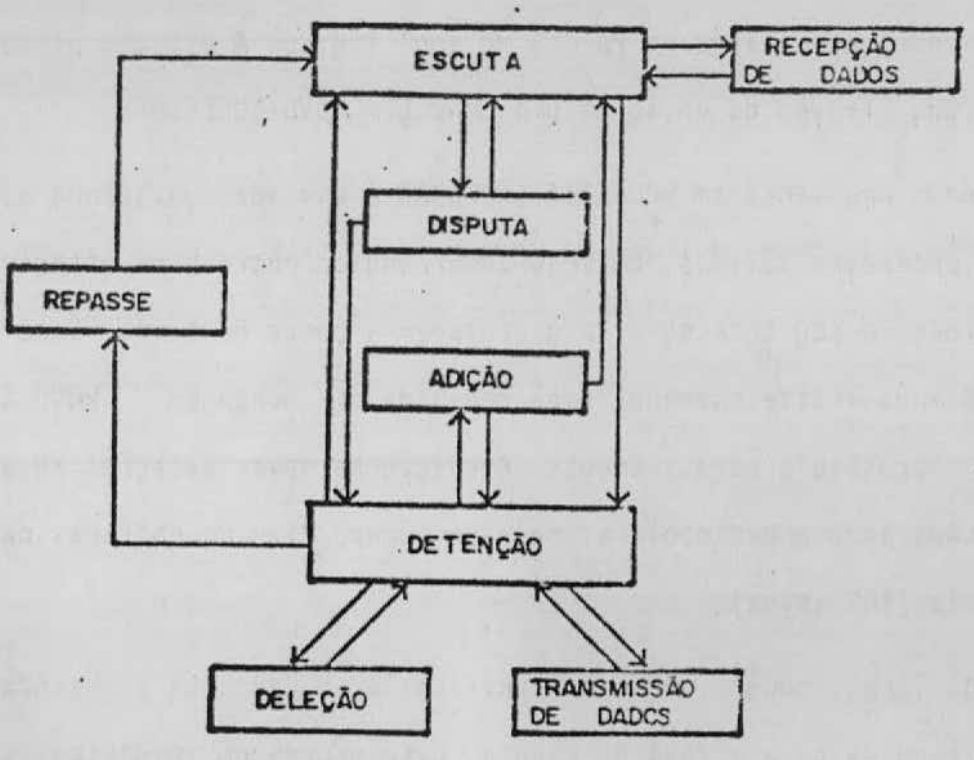

Figura 9 - Diagrama em Blocos do DIMAC

– Disputa pela Posse do Token, caso endereçado por uma mensagem SOLICITA-SUCESSOR (satisfeita a condição de seu endereço estar compreendido entre o da estação Origem e a Destino) ou ocorra Time-Out durante a fase de escuta, devido a prolongada inatividade do meio. Na primeira situação trata-se de uma adição de novas estações ao anel lógico, enquanto a segunda se refere a sua inicialização. Ao término desta fase de Disputa pela posse do Token, apenas uma estação resulta detentora do token, todas as demais retornam a fase de escuta. Uma mensagem NOVO-SUCESSOR será enviada pela estação que ficou com a posse do token, possibilitando assim a atualização do seu endereço no seu predecessor. No processo de disputa. as estações participantes transmitem concorrentemente mensagens DISPUTA-TOKEN cujo comprimento será função dos seus endereços.

Uma vez alcançada a fase de detenção uma estação poderá evoluir para as seguintes fases:

– Transmissão de Dados, onde mensagens oriundas da camada de controle de en-

lace lógico são enviadas;

- Deleção, onde uma estação se retira do anel lógico. A estação predecessora e comunicada através do envio de uma mensagem NOVO-SUCESSOR;
- Adição, onde uma mensagem SOLICITA-SUCESSOR é enviada convidando as estações com endereços válidos, ou seja compreendido entre o da estação detentora do token e seu sucessora, a disputarem a posse do token. Caso alguma estação atenda a este chamado, será recebida uma mensagem NOVO-SUCESSOR passando o protocolo para a escuta. A adição de novas estações ao anel será provocada após o protocolo atingir um número fixo de entradas na fase de detenção (100 vezes);
- Repasse do Token, onde o protocolo envia o Token para sua sucessora retornando em seguida para a fase de escuta. Caso ocorra um insucesso (sucessora fora da operação, por exemplo) o protocolo tenta repassar o token para a seguinte na sequência do anel lógico, através do envio da mensagem QUEM-SEGUE. Continuando não obtendo sucesso só restará evoluir para a fase de escuta, o que provocará indiretamente uma inicialização do anel.

3.2. Proposta de Especificação

Inicialmente vamos identificar a arquitetura hierárquica do protocolo de comunicação como composto de três camadas ou módulos (vide Figura 10): Camada Física, Camada de Controle de Acesso ao Meio (MAC) e a Camada de Controle de Enlace Lógico (LLC).

O nosso objetivo será a especificação do protocolo da camada 2 (MAC) da arquitetura apresentada, ou seja, o Protocolo de Controle de Acesso ao Meio. Na metodologia que propomos, identicamos a especificação de um protocolo como composta de cinco etapas distintas:

1. Caracterização das camadas envolvidas (etapa que acabamos de apresentar);
2. Especificação das interfaces utilizadas (utilizaremos FILAS com priorida-

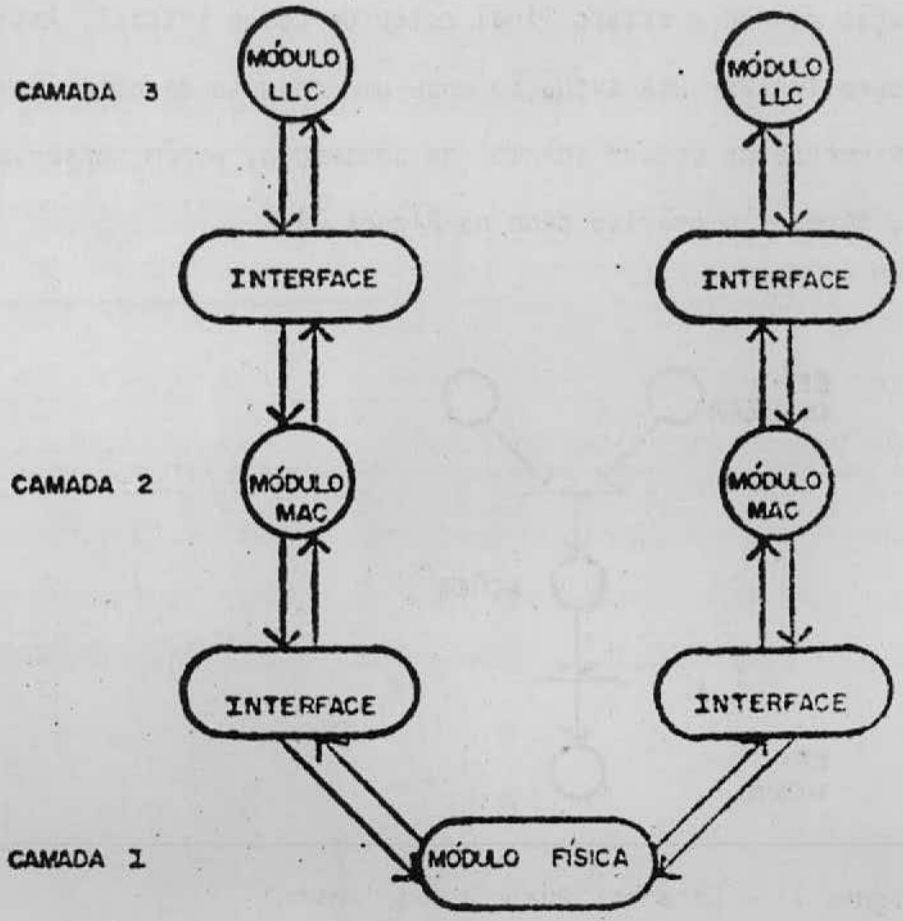

Figura 10 - Arquitetura de Camadas

de, devido a necessidade de envio prioritário de algumas primitivas de serviço entre camadas);

3. Identificação dos eventos (camadas por camada),
4. Especificação dos Estados do protocolo (de acordo com o diagrama em bloco do protocolo, Figura 9);
5. Especificação dos Conjuntos de Ações.

Durante a especificação serão definidas variáveis especiais chamadas "Estados". A execução de um grupo de ações decorrente da ocorrência de um evento está associada a um "Estado Inicial". Ao término de sua execução um no

vo estado, "Estado Final" será alcançado. Definiremos também uma variável chamada "Qualquer" que está relacionada aos estados do conjunto de "Estado Possíveis". De forma similar será definida a variável "Mesmo", usada para expressar a situação quando o estado final coincide com o inicial. Assim quando desejarmos caracterizar uma situação onde um conjunto de ações será executado independentemente do estado inicial do protocolo, porém conservando o estado inicial, teremos o gráfico dado na Figura 11.

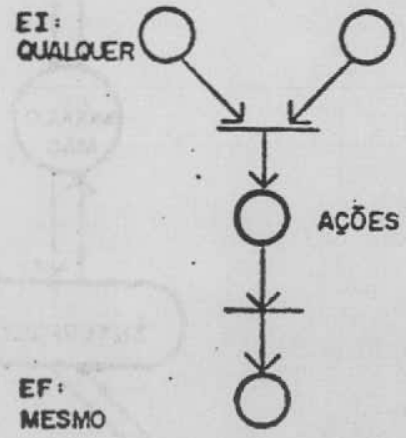

Figura 11 - Estados "Qualquer" e "Mesmo"

Um procedimento complexo normalmente será compactado em um único lugar. De fato este lugar servirá como abstração do diagrama em Rede de Petri associado. Por exemplo, o procedimento Envia X,Y, onde X se refere a uma determinada Fila e Y a uma mensagem, poderá ser representado por apenas um lugar ENV(x,y).

3.2.1. As Filas

As Filas, que modelam a presença de dados, constituem um dos primeiros ítens a serem definidos na especificação do Protocolo de Acesso ao Meio DIMAC. No total existem seis filas: duas são internas, duas compartilhadas com a camada superior (camada de enlace lógico) e duas da camada inferior

(física).

Filas Internas/Externas:

1. Transmissão - fila de mensagens a serem transmitidas através dos serviços das camadas inferiores (uso da primitiva "F-DATA.request");
2. Recepção - fila de mensagens recebidas através da primitiva "F-DATA.indication". Caso a mensagem seja de DADOS será posteriormente transmitida ao usuário (camada superior - LLC) através da primitiva M-DATA.indication;
3. SuperiorEntra - fila de primitivas do "Serviço de Controle de Acesso ao Meio", emitidas do uauário (camada superior - LLC) para a camada de controle de acesso ao meio;
4. SuperiorSai - fila de primitivas do "Serviço de Controle de Acesso ao Meio" emitida da camada de controle de acesso ao meio para o usuário (camada superior - LLC);
5. InferiorSai - fila de primitivas do "Serviço Físico", emitidas da camada de controle de acesso ao meio para a camada inferior (Física);
6. InferiorEntra - fila de primitivas do "Serviço Físico", emitida da camada física para a camada de controle de acesso ao meio.

As operações sobre as filas podem ser de quatro tipos: inserção no final da fila, remoção do primeiro elemento da fila, teste se vazia, inserção no início da fila. As filas compartilhadas entre duas camadas formam a "interface" entre elas. Por sua vez as operações sobre a interface são controladas pelo "Gerenciador de Interface".

3.2.2. Eventos

Como proposto na nossa metodologia, a programação de um módulo se baseia na ocorrência de eventos. No exemplo em consideração, estes podem ser originados pela Camada Superior (LLC), Camada Inferior (Física) ou internamente à própria camada (MAC).

Eventos originados pela Camada Superior (LLC):

MDADreq - presença de uma primitiva do tipo M-DATA.request na Fila SuperiorEntra;

MDELreq - presença de uma primitiva do tipo M-DELETE.request na Fila SuperiorEntra;

MPrimInv - presença de uma primitiva de um tipo diferente das aceitáveis (M-DATA.request, M-DELETE.request) na Fila SuperiorEntra;

Eventos originados pela camada Inferior (Física):

FDADind - presença de uma primitiva do tipo F-DATA.indication na Fila InferiorEntra;

FPrimInv - presença de uma primitiva do tipo diferente da aceitável (F-DATA.indication) na Fila InferiorEntra;

Eventos internos a própria camada (MAC):

RecTK - presença de uma mensagem do tipo TOKEN na fila Recepção;

RecQS - presença de uma mensagem do tipo QUEM-SEGUE na fila Recepção;

RecNS - presença de uma mensagem do tipo NOVO-SUCESSOR na fila Recepção;

RecSS - presença de uma mensagem do tipo SOLICITA-SUCESSOR na fila Recepção;

RecDT - presença de uma mensagem do tipo DISPUTA-TOKEN na fila Recepção;

RecDA - presença de uma mensagem do tipo DADOS na fila Recepção;

RecInv - presença de uma mensagem do tipo diferente de TK, QS, NS, SS, DT, DA, na fila Recepção;

Envia - presença de uma mensagem na fila de Transmissão. Corresponde a solicitação de Envio de Dados (M-DAT.request) oriunda da Camada Superior (LLC).

3.2.3. Estados do Protocolo

Segundo o diagrama em blocos do Protocolo (Figura 9), o conjunto de todos estados possíveis deve ser formado pelos seguintes componentes: Escu-

ta, Adição, Detenção, Disputa, Repasse, Repasse.Confirmação, Repasse.Quem-Segue.

Procurando facilitar a compreensão da especificação do DIMAC, bem como compactar os identificadores demasiadamentes extensos, adotaremos abreviações em protuguês para as primitivas existentes.

MDadReq	-	M-DATA.request
MDadInd	-	M-DATA.indication
MDadConf	-	M-DATA.Confirmation
MDelReq	-	M-DELETE.request
MDelInd	-	M-DELETE.indication
FDadReq	-	F-DATA.request
FDadInd	-	F-DATA.indication

3.2.4. Especificação dos Conjuntos de Ações

Levando em consideração o princípio de que o protocolo se comporta basicamente em função da presença de dados e do seu estado atual, procuramos identificar as fontes de dados bem como o estado inicial. As fontes serão as três camadas envolvidas: camada superior, a própria camada e a camada inferior. De acordo com o estado em que se encontra o protocolo e o tipo de mensagem recebida um conjunto de ações será ativado. Portanto a especificação pode ser dividida em cinco conjuntos (módulos):

1. Um evento e um conjunto de ações para cada tipo de primitiva que chega da camada superior através da fila SuperiorEntra (MDadReq, MdelReq), um evento e um conjunto de ações para consumir também qualquer tipo desconhecido de informação que esteja presente nesta fila, tratando-a como um erro (MPrimInv);
2. Um evento e um conjunto de ações para cada tipo de primitiva que chega da camada inferior através da fila InferiorEntra (FDadInd), um evento e um

conjunto de ações para consumir também qualquer tipo desconhecido de informação que esteja presente nesta fila, tratando-a como um erro (FPrimInv);

3. Um evento e um conjunto de ações para cada tipo conhecido de informação que chega na fila Recepção (RecTK, RecQS, RecNS, RecSS, RecDT, RecDA), e um evento e um conjunto de ações para consumir qualquer tipo desconhecido de informação (RecInv) que esteja presente nesta fila, invocando o procedimento de tratamento de erros;
4. Um evento e um conjunto de ações para tratar as mensagens que chegam na fila Transmissão, gerada em decorrência de procedimento tomado pelo protocolo da camada superior. Estas informações serão enviadas para a camada inferior através da primitiva de envio de dados (FDadReq);
5. Um conjunto de ações espontâneas ou seja, aquelas que não dependem da ocorrência de eventos para a sua execução.

A estrutura básica da especificação do protocolo seria portanto, composta de cinco coleções de subgrafos (módulos), cada coleção associada a um dos cinco conjuntos de ações identificados acima. Associado ao primeiro conjunto de eventos estão os diagramas apresentados nas Figuras 12 e 13 e ao segundo conjunto de eventos os diagramas mostrados nas Figuras 14. Uma descrição mais detalhada e envolvendo os outros conjuntos de eventos foge ao escopo deste trabalho e pode ser encontrada em [Castro 86].

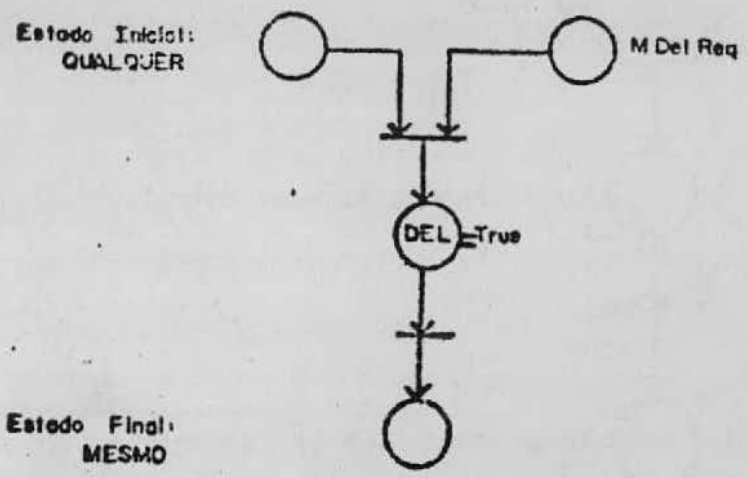

Figura 12 - Evento MDelReq

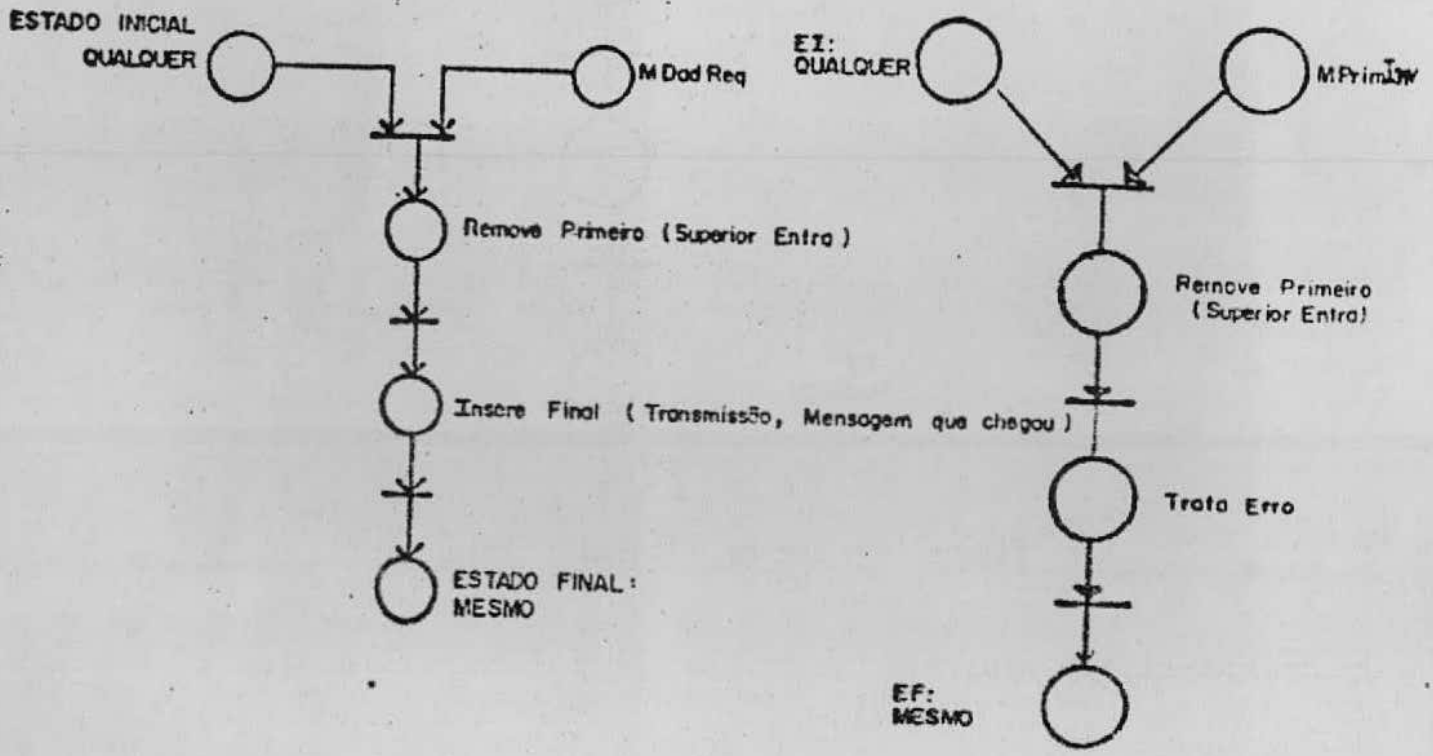

Figura 13 - Eventos MDadReq e MPrimInv

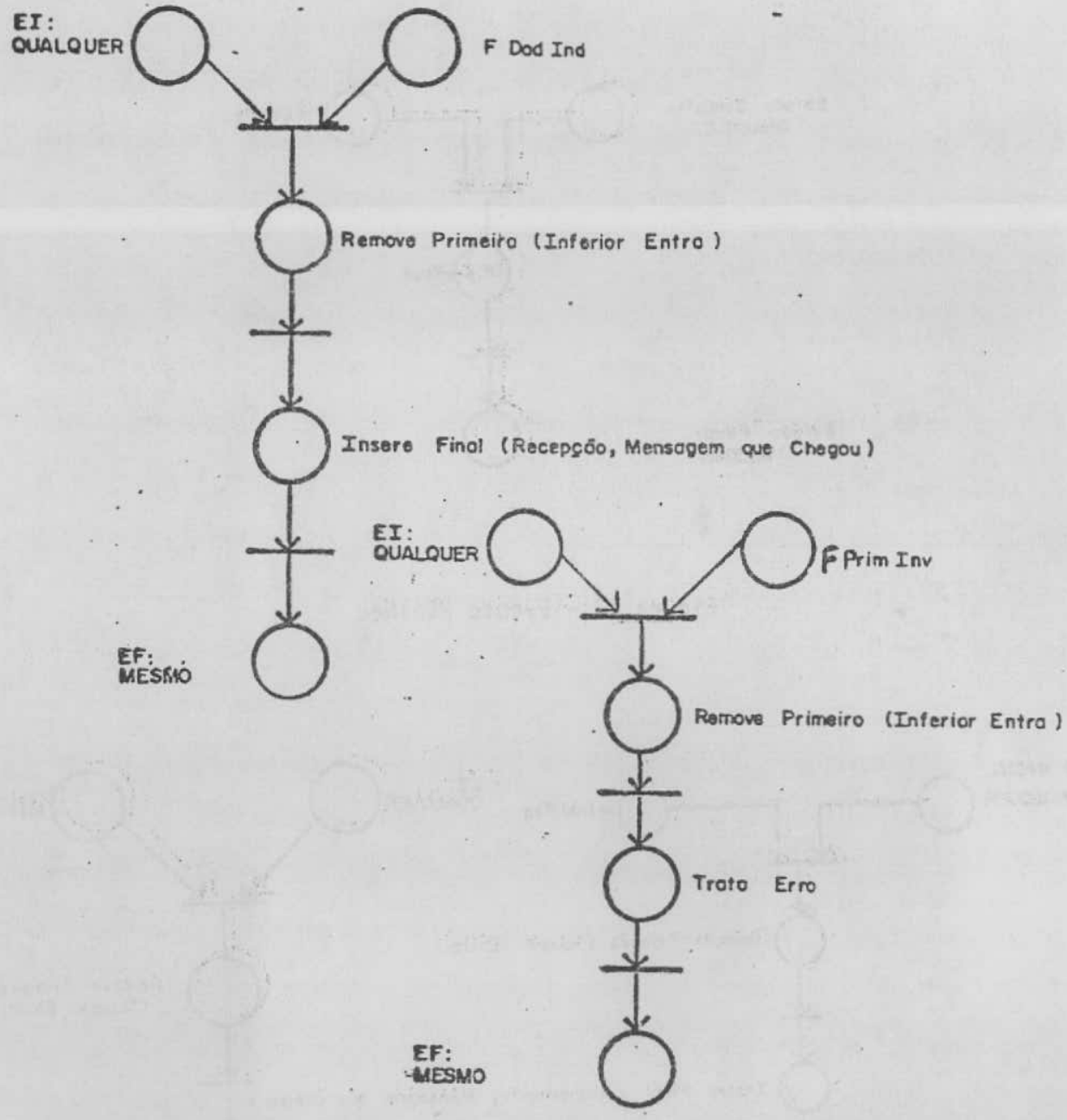

Figura 14 - Eventos FDadInd e FPrimInv

4. *CONSIDERAÇÕES FINAIS*

No presente trabalho propomos uma metodologia para especificação de arquiteturas hierárquicas de protocolos de comunicação entre computadores. Este objetivo foi alcançado através do uso das Redes de Petri Estendidas. Utilizamos alguns dos princípios básicos da Engenharia de Software tais como:

estratificação em camadas, modularidade e refinamento passo a passo.

De acordo com a metodologia apresentada, os módulos especificados capturam as funções básicas de cada camada e interagem de um modo transparente ao protocolo, através de Interfaces. Um módulo, se necessário, poderá ser refinado em submódulos.

Refletindo uma característica do ambiente dos protocolos de comunicação, adotamos o modelo de programação orientado para a ocorrência de eventos. A cada mensagem recebida por intermédio da Interface, associamos um evento. De acordo com estado do protocolo e do evento ocorrido um conjunto de ações é executado. Incorporamos aos conjuntos de ações, o conceito de prioridade e temporização, através de extensões às Redes de Petri clássicas.

No exemplo apresentado, especificação do Protocolo de Controle de Acesso ao Meio DIMAC, identificamos cinco tipos de conjuntos de ações: um conjunto correspondendo a chegada de primitivas originadas pela camada superior, outro correspondendo a chegada de primitivas originadas pela camada inferior, mais um para tratar os eventos decorrentes da fila Recepção (interna a própria camada), um conjunto de ações para tratar os eventos decorrentes da fila Transmissão (interna a própria camada) e finalmente um conjunto relativo aos eventos espontâneos.

5. *REFERÊNCIAS*

[BerTer 83] Berthelot,G.; Terrat,R.: "Petri Nets Theory for the Correctness of Protocols"; IEEE Transactions on Communications, vol. 30, número 12, Dezembro 1983, (pp. 2497-2513).

[Boch 80a] Bochmann,G.V.: "Formal Methods in Communications Protocol Design"; IEEE Transactions on Communications, vol. 28, nº 4, Abril 1980, (pp 624-631).

[Boch 80b] Bochmann,G.V.: "A General Transition Model for Protocols and Communications Services"; IEEE Transactions on Communications, vol. 28, nº 4, Abril 1980 (pp. 643-650).

[Castro 86] Castro,J.F.B.: "Uma Metodologia para Especificação de Protocolos Baseada em Redes de Petri"; Tese de Mestrado, Departamento de Informática da UFPE, Recife, Brasil, Agosto 1986.

[FeCaCu 86] Fernandes,C.G.; Castro,J.F.B., Cunha,P.R.F.: "DIMAC: Um Protocolo de Controle de Acesso ao Meio"; Anais do IV Simpósio Brasileiro de Redes de Computadores, Recife, Brasil, Março 1986, (pp. 199-217).

[Peters 81] Peterson,J.L.: "Petri Net Theory and Modelling of Systems"; Prentice-Hall Inc., Englewood Cliffs, New Jersey, Estados Unidos, 1981.

[QueCun 84] Queiroz,R.J.G.B.; Cunha,P.R.F.: "Implementação de Arquiteturas Hierárquicas de Protocolos"; Anais da IV Conferência Internacional en Ciencias de la Computacion, Santiago, Chile, Junho 1984.

[Queiroz 84] Queiroz,R.J.G.B.: "Uma Metodologia de Programação para Implementação de Protocolos"; Tese de Mestrado, Departamento de Informática, Universidade Federal de Pernambuco, Recife, Brasil, 1984, 172 pp.