2º SIMPÓSIO BRASILEIRO SOBRE REDES DE COMPUTADORES (2º SBRC)

UMA TÉCNICA PARA O DESENVOLVIMENTO E IMPLEMENTAÇÃO DE PROTOCOLOS

José Augusto Suruagy Monteiro

Departamento de Informática da UFPE
Centro de Ciências Exatas e da Natureza
Cidade Universitária
50.000 - RECIFE - PE

RESUMO

Este trabalho apresenta uma técnica para o desenvolvimento e implementação de protocolos. São apresentados o método de descrição adotado, e dois sistemas que visam dar suporte ao projeto e fornecer implementações do mesmo, além de outras ferramentas a serem desenvolvidas no futuro.

1. INTRODUÇÃO

Redes de Computadores apareceram como resposta à necessidade de se compartilhar recursos computacionais e informações distribuídas geograficamente.

Para isto é necessário que os computadores envolvidos interajam cooperativamente. O conjunto de regras que regula este "diálogo" é denominado de "protocolo de comunicação".

De modo a tornar o projeto destes protocolos o mais modular possível, a ISO ("International Standard Organization") apresentou uma arquitetura de camadas hierárquica, ou níveis, denominada de "Arquitetura para Conexões de Sistemas Abertos" [ZIMM80].

Uma vez definidas as funções de cada uma destas camadas, podem ser construídas implementações diferentes, fazendo uso de diversas tecnologias desde que se mantenham os serviços fornecidos por ela e as interfaces entre as camadas adjacentes.

A descrição de protocolos através de linguagens naturais (como o português) pode sempre deixar margem a dúvida, má interpretação e, conseqüentemente, ambigüidade.

O uso de descrições formais faz com que sejam reduzidas a possibilidade de erros, o risco de serem obtidas implementações inconsistentes, além de aumentar a clareza, compreensibilidade e facilidade a documentação [MERL76] [DANT78].

Os formalismos apresentados residem em três categorias principais [BOCH80]: modelos de transição de estados [BOCH78] [MERL76] [HARA78] [TENG80a] [TENG80b]. Linguagens de Programação [STEN76] [SCHW81] e métodos híbridos [SCHU80] [MENA81] [BLUM82].

A dificuldade de testar um protocolo em funcionamento enfatiza a importância de que o projeto do mesmo esteja correto [WEST78]. Daí a necessidade da utilização de técnicas de verificação do protocolo. Denomina-se VALIDAÇÃO à verificação de algumas propriedades gerais tais como *ausência de impasse*, *completeza*, *estabilidade* (*auto-sincronização*), *progresso e terminação própria* [TENG80b], [MONT82] [MONT83].

As técnicas utilizadas para a verificação de protocolos dependem do método empregado para descrevê-los. Para o método de transição de estados é usada a "análise de alcançabilidade" [DANT80] [MERL79] e para as linguagens de programação são utilizadas técnicas para a verificação de programas [BRAN78] [HAIL80].

Uma vez projetado o protocolo é preciso implantá-lo em cada uma das máquinas que compõem a rede. Com a utilização de siste-

mas que gerem implementações, ao menos parciais, de protocolos automaticamente, podemos incrementar a confiabilidade das mesmas (correspondem ao que está descrito) além de reduzir o esforço gasto na implementação de protocolos.

Protocolos consistem basicamente de estados e ações executadas em resposta à ocorrência de "eventos" tais como comando do nível superior, chegada de mensagens (do nível inferior) e fim de temporizações internas.

Esta observação leva-nos a utilizar o formalismo de máquinas de estados para especificar a operação de uma entidade de protocolo, onde estes eventos constituem as entradas [SUNS79] [CHUN78] [BLUM82] [SMIT83].

Além do mais, é preciso que alguém forneça um serviço de certificação de que uma dada implementação, desenvolvida por outros grupos, esteja de acordo com a especificação do protocolo. Foram desenvolvidos diversos *Testadores de Protocolos* com esta finalidade [BART80] [WEIR78] [PIAT80].

Todas estas etapas apresentadas anteriormente, isto é: descrição formal, verificação/validação, implementação e teste de implementações, constituem um processo que podemos denominar de "ciclo de vida do protocolo".

Diversos autores propuseram métodos e ferramentas de software visando automatizar o mais possível todas estas etapas. Sistemas que englobam a maior parte delas podem ser encontrados nas referências [BLUM82] e [SMIT83].

Este artigo visa apresentar o trabalho desenvolvido pelo autor [MONT82] para o desenvolvimento de sistemas de apoio ao projeto de protocolos e geração automática de protocolos, além de propor novas ferramentas a fim de completá-los.

2. *DESCRIÇÃO DE PROTOCOLOS ATRAVÉS DE GRAMÁTICAS*

Tomando como base os formalismos já existentes para a descrição de protocolos, procurou-se escolher um que se adaptasse bem a ser utilizado dentro de um "ciclo de vida" completo do protocolo. Para isto o método deveria satisfazer os seguintes requisitos [MONT82]: i) fornecer uma descrição precisa, ii) ser facilmente compreendida pelo usuário, iii) ser facilmente compreendida pelo computador, iv) possuir uma estrutura de níveis e, finalmente, v) facilitar a validação e a geração de implementações automaticamente.

Dentre os formalismos analisados aquele que se mostrou mais adequado, inclusive pela disponibilidade de ferramentas já desenvolvidas foi a Gramática de Transmissão proposta por Teng em [TENG80a].

A motivação do uso de "Linguagens Formais" (Gramáticas) provém da correspondência existente com a máquina de estados. Cada sentença gerada pela gramática corresponderá a uma sequência de ações válida do protocolo.

Como veremos a seguir, as gramáticas podem descrever tanto as ações como os formatos das mensagens.

Os trabalhos significativos nesta área foram apresentados por Harangozó [HARA78] e Teng em [TENG80a] e [TENG80b].

A notação utilizada para descrever uma gramática de transmissão é a mesma utilizada na teoria de linguagens formais [HOPC69], onde uma gramática é representada pela quádrupla G=(Vn,Vt,P,<S>) onde:

Vn é um conjunto de ações ou de mensagens não terminais, envolvidas pelos símbolos "< >";

Vt é um conjunto de símbolos de ação ou de mensagem terminais;

P é um conjunto de regras de produção onde cada elemento é da forma <A> ::=X, onde A ε Vn e X ε (Vn $\cup$ VT)*

S ε Vn é o símbolo inicial.

As regras de produção também podem ser escritas na forma:

não-terminal::=alternativa-1 |alternativa-2|...|
alternativa-n (n>0)

Uma alternativa pode ser qualquer sequência de terminais e não-terminais, desde que representem sequências de ações distintas mas igualmente válidas do protocolo.

Os símbolos não terminais representam estados ou reúnem sequências de ações a serem executadas e que estão detalhadas através de outras produções.

Não-terminais que representam estados são aqueles cujas derivações são efetuadas unicamente após a ocorrência de um dado evento. Estes eventos podem ser externos como recebimento de mensagens, ou internos, como estouro de temporização.

Uma produção válida da gramática de ações poderia ter o seguinte aspecto:

<estado_1>::=<ações_1> <estado_2>|
<ações_2> <estado_1>.

O não-terminal à esquerda da produção representa o estado a partir do qual podem ser disparadas duas sequências de ações, a partir de eventos distintos para que a gramática (protocolo) seja determinística. Ao final de uma sequência de ações disparadas por um evento, atinge-se um outro não-terminal que representa um estado. A derivação deste não-terminal numa sequência de ações só será efetuada quando da ocorrência de um novo evento. O ponto indica o final da última alternativa.

Os símbolos terminais no estágio final de detalhamento do protocolo representam rotinas de ações semânticas, ou seja, a elas estão associadas operações elementares ou *primitivas* que podem ser divididas basicamente em cinco grupos:

a. ações de gerenciamento de memória
b. ações de recepção e de transmissão
c. ações de segmentação e montagem de mensagens
d. ações de sincronização de eventos
e. ação de teste condicional (CND)

No primeiro grupo estão incluídas as ações de "alocação' (ALC) e "Liberação" (LIB) de "buffers".

Nas ações de recepção e de transmissão, incluem-se as de "transmissão" (TXT) e "recepção" (RCB) de mensagens.

Na categoria de ações de sincronização de eventos, encontram-se as ações de manipulação de filas: coloca na fila (COL), retira da fila (RET), olha a fila (OLH) e limpeza da fila (LMP); assim como a de disparo (LGT) e desligamento de temporizadores (DLG).

Outras ações representam operações elementares tais como: atribuição de valores a variáveis (ATR), incrementar variável (INC), decrementa variável (DEC), subtração (SUB).

Cada terminal é representado do seguinte modo: três caracteres são utilizados para identificar a rotina que está sendo chamada (apenas letras); o caractere seguinte é um ponto que serve para separar o identificador dos parâmetros que lhe estão associados. O número de parâmetros depende da rotina mas devem ser separados através de vírgulas. Para identificar que se trata de um símbolo terminal, este é envolvido por aspas.

Algumas destas rotinas podem ser rotinas-padrão que independem do protocolo específico, enquanto que outras podem possuir procedimentos específicos daquele protocolo.

Algumas das rotinas são dependentes da máquina específica e neste caso deverá ser fornecida uma diferente cada vez que se fi

zer a implementação para uma outra máquina, apesar de se tratar do mesmo protocolo.

As ações características ao protocolo fazem parte da sua descrição, enquanto que as independentes do protocolo podem estar já descritas num arquivo de biblioteca.

As ações dependentes da máquina, chamadas diretamente da gramática de ações ou chamadas dentro da descrição de uma outra ação, deverão ser escritas em assembler ou numa linguagem de alto-nível compilável na máquina em questão.

Após o cabeçalho de uma ação dependente de máquina, segue-se a palavra reservada "EXTERNA". Logo após o cabeçalho de uma ação que se encontra na biblioteca, deve ser colocada a palavra reservada "BIBLIOTECA".

As ações semânticas devem ser escritas em LIP (Linguagem Intermediária de Programação) [MONT82] que é uma linguagem projetada de modo a permitir a tradução direta de suas construções nas construções de outras linguagens de programação (Fortran, Algol, Pascal, PL/I, etc) através de um macro-expansor genérico.

A *Gramática de Mensagens* é similar à gramática de ações. Nela ao invés de termos símbolos não-terminais representando estados ou reunindo sequências de ações a serem executadas, estes representarão mensagens ou sequências de campos das mesmas.

Os símbolos terminais ao invés de representarem chamadas de rotinas de ações semânticas, correspondem à descrição precisa do campo, que inclui o tipo, comprimento e variáveis ou valores associados, se for o caso.

Para que os símbolos terminais contenham as informações citadas anteriormente, foram definidos inicialmente 5 tipos de terminais: 2 para representar campos de comprimento fixo (com conteúdo variável (V) ou constante (C), respectivamente) e 3 para representar campos de comprimento variável (como campo de dados (D), campo variável dependente (VD) - cujo comprimento, depende do valor de um campo anterior - e completa unidade de campo (CC), utilizada em conjunto com campos do tipo variável dependente).

Para fazer a distinção entre mensagens de nível e mensagens de interface com os níveis superior e inferior são utilizados respectivamente, os seguintes identificadores: ∂N, ∂N+1 e ∂N-1, que devem preceder os identificadores das mensagens de cada uma destas categorias.

Para cada uma delas podemos ter mensagens que são apenas recebidas, outras que só são transmitidas e outras que tanto podem

ser transmitidas como recebidas (pela mesma entidade). Para identificá-las deveremos fazer preceder aos não terminais que identificam as mensagens de um dado tipo, respectivamente os seguintes delimitadores: %RX,%TX e %DX ("Duplex" - "Default").

Descrição Completa do Protocolo

A Descrição Completa de um protocolo envolve desde o formato das mensagens trocadas com os níveis adjacentes e com a entidade parceira, às Ações Semânticas associadas a seu comportamento e, finalmente, ao comportamento da entidade de nível diante da ocorrência de algum dos eventos possíveis.

A ordem em que devem ser apresentados estes elementos é justamente a acima, isto é, Gramática de Mensagens, Ações Semânticas, e Gramática de Ações.

Onde cada um deles é precedido por um delimitador específico, ou seja:

```
<Descrição_Completa>::= #MENSAGENS
                        <Gramatica_de_mensagens>
                        #AÇOES_SEMANTICAS
                        <Sequencia_de_açoes>
                        #GRAMATICA_DE_AÇOES
                        <Gramatica_de_açoes>
                        <FIM>

<FIM> :: #FIM|#
```

Exemplo

Como exemplo do uso da Gramática de Transmissão apresentada anteriormente, descreveremos o protocolo do bit alternante [BART69]. Este protocolo foi projetado de modo a prover uma transmissão full-duplex confiável, em linhas half-duplex. Cada mensagem transmitida por uma entidade possui informações para a deteção de erros e um bit de controle denominado de "bit alternante". Quando for transmitida uma mensagem com bit de controle 0, a proxima mensagem com bit de controle 1 deverá ser transmitida apenas quando o reconhecimento da mensagem anterior tiver sido recebido.

Este protocolo possui basicamente dois estados, o estado PRONTO e o estado ESPERA-REC. Na figura 2.1 é apresentado o diagrama de transição de estados correspondente. Cada transição indicada por uma flecha-corresponde a um evento indicado em cima da linha e pode provocar uma saída, que está indicada abaixo da linha.

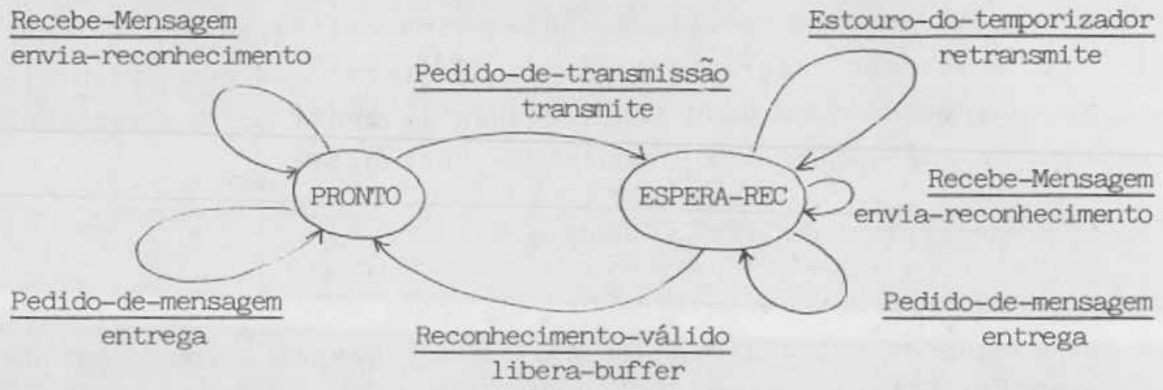

FIG. 2.1 Diagrama de Transição de Estado para o protocolo do "bit alternante".

A entidade encontra-se inicialmente no estado PRONTO e ao receber do nível superior um pedido de transmissão de mensagem, ela é copiada num buffer para possível retransmissão, dispara-se o temporizador (para limitar o intervalo de espera por reconhecimento), a mensagem é transmitida, e a entidade entra no estado ESPERA-REC. Se o temporizador estourar estando a entidade ainda no estado ESPERA-REC, a mensagem é recuperada do buffer e retransmitida.

Estando no estado ESPERA-REC, ao receber um reconhecimento válido da entidade parceira, libera o buffer de retransmissão, desliga o temporizador e retorna ao estado PRONTO.

Se em qualquer dos estados ocorrer a recepção de uma mensagem ela será reconhecida e será guardada caso corresponda ao número de sequência (bit de controle) esperado. A entidade permanece no estado em que se encontrava.

Se o nível superior solicitar alguma mensagem, o seu pedido será atendido caso haja alguma mensagem recebida.

Na Descrição Formal do Protocolo que segue, comentários estarão envolvidos pelos delimitadores (* e *).

```
#MENSAGENS   (* GRAMATICA DE MENSAGENS *)

@N+1         (* INTERFACES COM O NIVEL SUPERIOR *)
%RX          (* APENAS RECEBIDAS - COMANDOS *)

<PEDE_TX>     ::= C,1BY,H'0' <DADOS> .   (* PEDIDO DE TRANSMIS-
                  SAO DE MENSAGEM - COMPOE-SE DE UM CAMPO
                  QUE IDENTIFICA A INTERFACE, E DO  CAMPO
                  DE DADOS  *)
```

```
<DADOS>       ::= D,...BY, .    (*  AS RETICENCIAS ESTAO INDI-
                                  CANDO QUE O COMPRIMENTO MA-
                                  XIMO DO CAMPO NAO ESTA' DE-
                                  FINIDO  *)
<PEDE_MSG>    ::= C,1BY,H'1' .  (* PEDIDO DE MENSAGEM RECEBIDA -
                                  BASTA A SUA IDENTIFICACAO  *)

%TX           (* APENAS TRANSMITIDA - RESPOSTA *)

<MSG_REC>     ::= C,1BY,H'0'  <DADOS> .   (* ENTREGA MENSAGEM
                                              RECEBIDA    *)

@N            (* MENSAGENS DE NIVEL - TANTO PODEM SER TRANSMITIDAS
                 COMO RECEBIDAS  *)

<MENSAGEM>    ::= <IDENT_MSG> V,1B,NO_SEQ_R,MSG_TX <DADOS> .
                  (* MENSAGEM - COMPOE-SE DO IDENTIFICADOR, NO.
                     DE SEQUENCIA E DOS DADOS PROPRIAMENTE DITOS.
                     O TERMINAL INDICA QUE O NO. DE SEQUENCIA E'
                     UM CAMPO DE 1 BIT, DE CONTEUDO VARIAVEL,
                     QUE E' ATRIBUIDO A NO_SEQ_R NA RECEPCAO, E
                     E' COPIADO DE MSG_TX NA TRANSMISSAO  *)

<IDENT_MSG>   ::= C,1B,B'0' .

<REC>         ::= <IDENT_REC> V,1B,NO_SEQ_R, .
                  (* RECONHECIMENTO *)

<IDENT_REC>   ::= C,1B,B'1' .

#ACOES_SEMANTICAS

(*  APOS A PALAVRA CHAVE "ACAO" SEGUE O IDENTIFICADOR DA MESMA,
    UMA LISTA DOS PARAMETROS E UMA LISTA DOS TIPOS CORRESPONDEN-
    TES. DIVERSOS PARAMETROS APARECEM COMO SENDO DO TIPO "INTE-
    GER" DEVIDO 'A CODIFICACAO QUE E' FEITA. POR SIMPLICIDADE ,
    ASSUME-SE QUE TODAS AS ACOES ENCONTRAM-SE NA "BIBLIOTECA".
    NATURALMENTE ALGUMAS ACOES COMO "TXT", "LGT", "DLG" E "LIB"
    SAO ESPECIFICAS A CADA MAQUINA E DEVERIA SER FORNECIDA     *)

ACAO TXT NOME,DESTINO  MENSAGEM,NIVEL                          BIBLIOTECA
ACAO COL FILA  INTEGER                                         BIBLIOTECA
ACAO RET FILA  INTEGER                                         BIBLIOTECA
ACAO OLH OK,FILA  BOOLEAN,INTEGER                              BIBLIOTECA
ACAO LGT TEMP,DUR,UNID  INTEGER,INTEGER,INTEGER                BIBLIOTECA
ACAO DLG TEMP  INTEGER                                         BIBLIOTECA
ACAO INC VALOR,MODULO  INTEGER,INTEGER                         BIBLIOTECA
ACAO LIB                                                       BIBLIOTECA

#GRAMATICA_DE_ACOES

<PRONTO>              ::= <PEDIDO_DE_TRANSMISSAO> <TRANSMITE>
                              <ESPERA_REC> ,
                          <RECEBE_MENSAGEM>
                              <ENVIA_RECONHECIMENTO> <PRONTO> ,
                          <PEDIDO_DE_MENSAGEM>
                              <ENTREGA_SE_HOUVER> <PRONTO> .
```

```
<ESPERA_REC>          ::= <ESTOURO_DO_TEMPORIZADOR>
                              <RETRANSMITE> <ESPERA_REC> ,
                          <RECEBE_MENSAGEM>
                              <ENVIA_RECONHECIMENTO>
                              <ESPERA_REC> ,
                          <PEDIDO_DE_MENSAGEM>
                              <ENTREGA_SE_HOUVER> <ESPERA_REC> ,
                          <RECEBE_RECONHECIMENTO> <RESULTADO> .

<RESULTADO>           ::= <RECONHECIMENTO_VALIDO>
                              <LIBERA_BUFFER> <PRONTO> ,
                          <RECONHECIMENTO_INVALIDO>
                              <ESPERA_REC> .

<PEDIDO_DE_TRANSMISSAO> ::= "RCB.<PEDE_TX>,NS" .

<TRANSMITE>           ::= "COL.MSG_PEND" "LGT.TEMP,...,SG"
                              "TXT.<MENSAGEM>" .

<RECEBE_MENSAGEM>     ::= "RCB.<MENSAGEM>" <GUARDA_SE_ESPERADA> .

<GUARDA_SE_ESPERADA> ::= "CND.NO_SEQ_R,MSG_ESP,EQ"  (* SE O NO.
                              DE SEQUENCIA DA MENSAGEM RECEBIDA
                              FOR IGUAL (EQ) AO DA ESPERADA ,
                              ENTAO ... *)

                              "COL.MSG_REC" "INC.MSG_ESP,2" ,

                          "CND.NO_SEQ_R,MSG_ESP,NE" . (* CASO
                              CONTRARIO NAO FAZ NADA *)

<ENVIA_RECONHECIMENTO> ::= "TXT.<REC>" .

<PEDIDO_DE_MENSAGEM> ::= "RCB.<PEDE_MSG>,NS" .

<ENTREGA_SE_HOUVER>   ::= "OLH.HA_MSG,MSG_REC"
                              <RESULTADO_ENTREGA> .

<RESULTADO_ENTREGA>   ::= "CND.HA_MSG" "RET.MSG_REC"
                              "TXT.<MSG_REC>,NS" ,
                          "CND.HA_MSG,NT" .

<ESTOURO_DO_TEMPORIZADOR> ::= "EST.TEMP" .

<RETRANSMITE>         ::= "RET.MSG_PEND" <TRANSMITE> .

<RECEBE_RECONHECIMENTO> ::= "RCB.<REC>" .

<RECONHECIMENTO_VALIDO>  ::= "CND.NO_SEQ_R,MSG_TX,EQ" .
                             (*  TESTA SE O NO. DE SEQUENCIA DO
                                 RECONHECIMENTO RECEBIDO E' IGUAL
                                 AO DA ULTIMA MENSAGEM TRANSMITI-
                                 DA  *)

<RECONHECIMENTO_INVALIDO> ::= "CND.NO_SEQ_R,MSG_TX,NE" .

<LIBERA_BUFFER>       ::= "DLG.TEMP" "RET.MSG_PEND" "LIB"
                              "INC.MSG_TX,2" .

#FIM
```

3. *VALIDAÇÃO DE PROTOCOLOS*

Teng em [TENG80b] apresenta um sistema de validação que verifica algumas propriedades gerais citadas anteriormente, através da análise das seguintes características:

1. reconhecimento sem memória
2. GT bem estruturada
3. impasse de recepção
4. incompatibilidade
5. ciclos

Reconhecimento sem memória - Para o uso de técnicas de implementação automática necessitamos que todas as sentenças da GT possam ser reconhecidas sem que tenhamos necessidade de retornar a símbolos analisados anteriormente.

Para que as sentenças de uma gramática possam ser reconhecidas "sem memória", ele deve obedecer a duas restrições: ser determinística e não possuir recursões à esquerda.

GT *bem estruturada* - Para que a GT seja bem estruturada é necessário que não haja nenhum não-terminal não definido ou produções supérfluas, e que cada sentença (sequência de ações) termine propriamente.

A restrição de que a GT gere apenas sentenças próprias requer a terminação própria de todas as sequências de ações de uma GT. Esta propriedade é importante pois devemos estar certos que a partir de qualquer estado da GT é possível alcançar um estado final apropriado.

Um *impasse estático* ocorre quando a partir de um estado não se atinge um estado final apropriado. Uma GT bem estruturada tem a propriedade desejável de nunca chegar a um impasse estático.

Impasse de Recepção - impasses de recepção ocorrem quando uma entidade permanece num estado inativo aguardando a recepção de uma mensagem que por algum motivo (perda de mensagens, etc) nunca será recebida.

O mecanismo de recuperação deste tipo de impasse é através do uso de temporização. Devemos portanto ter ações de estouro de temporização e retransmissão como uma alternativa da produção.

Incompatibilidade - Este conceito está relacionado com a inexistência de transmissão por uma entidade de uma mensagem que a entidade parceira deseja receber ou da transmissão por parte de uma entidade, de uma mensagem para a qual não existe uma recepção

prevista por parte da entidade parceira.

Ou seja, deveria haver uma correspondência biunívoca entre as mensagens que podem ser transmitidas e as mensagens que se prevê sejam recebidas.

Ciclos (ou oscilações) - Estamos interessados em detectar todos os possíveis ciclos para analisar se estes são autolimitados e assim evitar a ocorrência de impasses dinâmicos (repetição contínua de uma sequência de ações sem que haja progresso na comunicação).

De modo a evitar uma possível sequência infinita de repetições (ciclos), uma das ações terminais do ciclo deverá limitar o número máximo de repetições do mesmo. Por exemplo, num ciclo do tipo:

<ESPERA_REC>::=<Estouro_do_Temporizador> <Retransmite>
<ESPERA_REC>

deve-se limitar o número máximo permitido de retransmissões, e esta limitação deverá estar presente na GT.

Em alguns casos esta limitação está implícita nos comandos do nível superior e portanto não precisa estar presente na GT deste nível.

VALIDAÇÃO GLOBAL

Até o momento tratamos da validação local do protocolo, isto é, analisamos a existência de impasses, etc, apenas do ponto de vista de uma entidade. Para que possamos ter um grau maior de confiança de que o protocolo funciona corretamente, deveríamos considerar também o estado da entidade parceira e o do meio de comunicação.

Teng apresenta em [TENG80a] o modelo de um autômato de validação que é facilmente derivável a partir da GT. Para contornar o problema da "explosão de estado" resultante desta abordagem global, ele apresenta diversas regras de redução, mas ainda não existem algorítmos que permitam a aplicação automática destas regras de redução. Uma opção é desenvolver um sistema que facilite a análise e a aplicação destas regras por parte do projetista.

O SAPPRO (SISTEMA DE APOIO AO PROJETO DE PROTOCOLOS)

O SAPPRO [MONT82] e [MONT83] compõe-se de diversas ferramentas com o intuito de, trabalhando cooperativamente com o projetista, auxiliá-lo na validação, correção e refinamento da descrição

formal que lhe é fornecida.

O ciclo completo do projeto de protocolos está representado na figura 3.1.

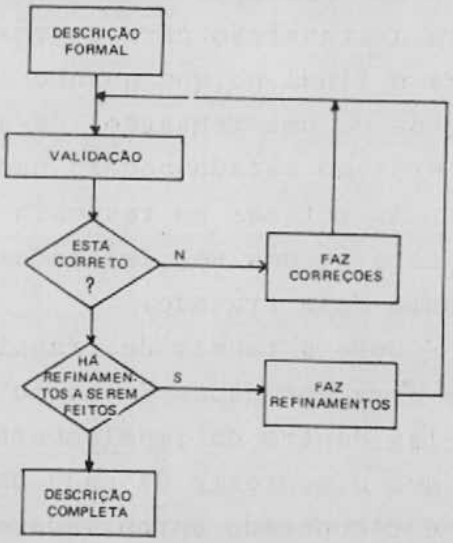

Figura 3.1. CICLO DO PROJETO DE PROTOCOLOS

4. IMPLEMENTAÇÃO DE PROTOCOLOS

Conforme visto na introdução, a observação de que o comportamento de um protocolo corresponde à execução de ações em resposta à ocorrência de eventos leva-nos a utilizar o formalismo de máquinas de estado.

Na realidade, cada entidade de um certo nível corresponde a uma máquina de estados finita. Este formalismo é usado em [CHUN78] [BLUM82] e [SMIT83].

Podemos visualizar uma realização da Máquina de Estados como sendo composta das seguinte etapas:

a) *identificação do evento* - antes de mais nada é preciso identificar o evento que será tratado, isto é, que temporizador estourou, ou que tipo de mensagem foi recebida e quem a enviou (o nível superior ou o nível inferior).

b) *seleção da Máquina de Estados a que se refere* - após a identificação do evento, deve-se identificar a que entidade se refere para que seja selecionada a Máquina de Estados correspondente.

c) *consulta à Tabela de Transição* - uma vez identificada a máquina de estados, pode-se recuperar o seu estado atual e assim, através do par estado atual/evento, recuperar da Tabela de Transição, o rótulo da rotina de ações a serem executadas.

d) *execução das rotinas correspondentes* - munidos do rótulo, chama-se o procedimento correspondente que corresponde a uma série de ações tais como transmissão de mensagens, manipulação de variáveis, disparo de temporizadores, etc.

e) *transição de estado* - ao final da execução das rotinas é feita a transição para o próximo estado. Esta transição é deixada para o final porque quando da ocorrência de um dado evento como a chegada de uma mensagem, de acordo com o valor de um de seus campos o próximo estado poderá não ser o mesmo.

As rotinas em resposta a ocorrência de eventos são "atômicas", isto é, uma vez iniciadasvão até o seu final antes que um novo evento seja tratado.

Como a tabela de Transição e as Rotinas de ações são as mesmas para entidades de mesmo tipo, não há necessidade de multiplicá-las dentro da impelementação de um dado nível de protocolo. Basta que o contexto de cada uma delas (estado e parâmetros) seja salvo e recuperado apropriadamente.

MODELO DE IMPLEMENTAÇÃO

Na figura 4.1 encontra-se o modelo de implementação proposto em [MONT82].

Neste modelo. qualquer mensagem destinada às entidades do nível assim como as indicações de estouro de temporizador são colocadas na Fila de Entrada. O gerenciador da Fila é o módulo encarregado de retirar elementos da fila, possivelmente usando algum critério de prioridade.

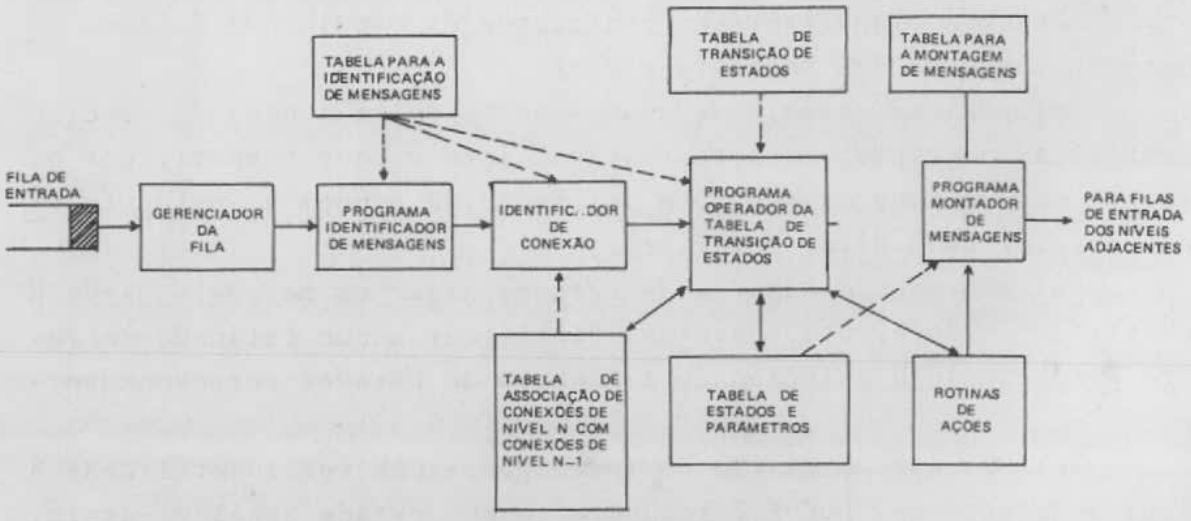

Figura 4.1. MODELO DE IMPLEMENTAÇÃO DE PROTOCOLOS

As mensagens recebidas devem ser identificadas através do reconhecimento de padrões de bits e comprimentos válidos. Esta é a função do Programa Identificador de Mensagens (PIM) que para isto utiliza-se de uma tabela que nada mais é senão a Tabela de Transição de Estados do Autômato Reconhecedor de Mensagens.

Após a identificação da mensagem, caso haja mais de uma possível conexão no nível (isto é, haja mais de uma entidade) é preciso identificar a que conexão a mensagem se refere. Isto é feito através do conteúdo de um determinado campo da mensagem. Para o caso de protocolos que fazem multiplexação de conexões do nível inferior, é preciso também consultar uma "Tabela de Associação de Conexões de Nível N com conexões de nível N-1.

Uma vez identificadas a mensagem e a conexão a que se refere, estas informações são passadas ao POTE (Programa Operador da Tabela de Transição de Estados) juntamente com os demais campos da mensagem, que serão utilizados durante a execução da rotina.

Inicialmente, a partir do identificador da conexão e através de uma consulta à tabela de estados e parâmetros (TABPARAM), é recuperado o estado corrente da conexão em questão.

De posse do estado corrente e da identificação da mensagem (evento ocorrido) pode ser feita a consulta à Tabela de Transição de Estados (TABTRANS). Deste modo, obtém-se o rótulo inicial da rotina que deverá ser executada em resposta a este evento.

Pode-se então iniciar a execução das rotinas. Será a execução das ações semânticas associadas que irá provocar a atualização dos parâmetros da TABPARAM, o pedido de Transmissão de Mensagens ao Programa Montador de Mensagens (PMM) e ao final, se houve uma transição de estado, atualizar o estado para esta conexão na TABPARAM.

Para montar as mensagens, o PMM faz consultas a uma Tabela que contém um a um a identificação dos diversos campos de cada uma das mensagens. Cada uma destas entradas contém informações tais como tipo do campo, comprimento, valor (no caso de campo constante) e identificação da variável cujo conteúdo deverá ser copiado no campo, etc.

GERAÇÃO AUTOMÁTICA DE IMPLEMENTAÇÕES

A geração automática de implementações é interessante por facilitar o trabalho de implantação e manutenção de protocolos numa rede, em particular se os computadores envolvidos forem na sua

maioria, heterogêneos.

Segundo [SMIT83] os métodos atuais para implementação direta podem ser utilizados apenas quando o custo do desenvolvimento ou restrições de tempo forem mais importantes do que a otimização do desempenho.

Implementações obtidas automaticamente provaram ser muito úteis em testes e experimentos, e alguma delas podem se tornar produtos finais [BLUM82].

De qualquer modo estas implementações podem ser ao menos o "esqueleto" da implementação final.

Voltando ao modelo de implementação apresentado, observa-se claramente que existe um conjunto básico de programas comuns a qualquer protocolo. São eles: o Gerenciador da Fila, o Programa Identificador de Mensagens, o Identificador de conexão, o Programa Operador da Tabela de Transição de Estados e o Programa Montador de Mensagens. E verifica-se também que existem tabelas que contêm as características particulares de um dado protocolo. São elas: a Tabela para a Identificação de Mensagens, a Tabela de Associação de conexões do nível N com conexões de nível N-1, Tabela de Transição de Estados, Tabela de Estados e Parâmetros e Tabela para a montagem de mensagens. É característico também de um dado protocolo, o módulo que contém as Rotinas de Ações.

A idéia básica que existe por trás do Sistema Automático de Geração de Protocolos (SAGEP) proposto em [MONT82] é justamente o de montar as tabelas e construir as rotinas de ações a partir da descrição (particular) do protocolo.

5. OUTRAS FERRAMENTAS

Além das apresentadas anteriormente, diversas "ferramentas" de software podem ser desenvolvidas de modo a facilitar o desenvolvimento e teste de protocolos.

SIMULADOR

Uma delas seria a utilização das tabelas e rotinas construídas pelo SAGEP por um sistema que os controlasse de modo a permitir ao projetista observar o comportamento do protocolo após o envio de alguns comandos.

Um sistema deste tipo poderia ter o aspecto do apresentado na Fig. 5.1. O controlador, através de comandos do projetista geraria comandos para uma das entidades do nível, controlaria as filas

que modelam o meio de comunicação – de modo a permitir um comportamento semelhante ao esperado do nível inferior. Além do mais poderia acompanhar o estado das conexões de cada entidade de nível e das filas e, inclusive, alterar o estado, "perder" mensagens, duplicá-las, modificar a ordem, etc.

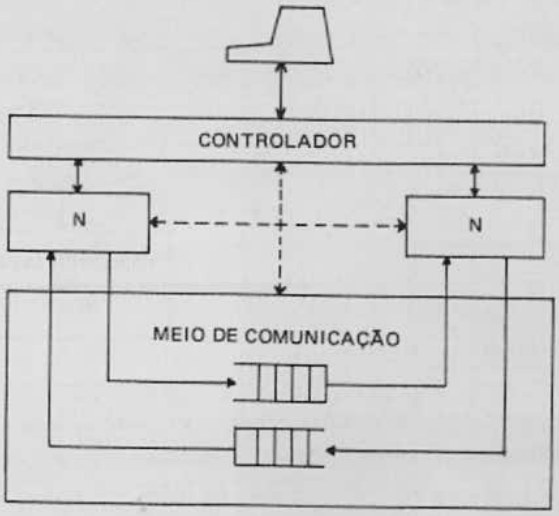

Figura 5.1. SIMULADOR DE PROTOCOLO

Pode haver também dois modos de funcionamento: o manual e o automático. No modo manual, caberia ao projetista pedir serviços, etc; no modo automático o próprio controlador poderia gerar comandos, perder mensagens de acordo com uma certa probabilidade, analisar as respostas e fornecer ao projetista um relatório dos "sintomas" observados.

Se desejarmos esgotar todos os possíveis estados para verificar se o comportamento é o desejado, de acordo com a capacidade de controle e/ou observação, o comprimento do teste pode variar de n^3 a n, onde n é o número de estados [PIAT80]. O pior caso seria o de controlar a entrada e observar a saída, e o melhor caso seria o que pudéssemos estabelecer e observar o estado em paralelo, além de controlar as entradas e observar as saídas.

Um esquema deste tipo foi utilizado pela IBM [SMIT83].

TESTADOR DE IMPLEMENTAÇÕES

Se as implementações de protocolos forem todas obtidas através de um sistema automático, a partir de uma mesma descrição formal, podemos garantir que se o sistema não gerar erros as implementações por ele derivadas são altamente confiáveis.

No entanto, na implantação de uma rede em geral grupos dis

tintos constroem suas implementações independentemente, fazendo-se portanto necessária a existência de entidades que expeçam certificados atestando a conformidade das mesmas com a especificação [BART80] [WEIR78].

Um testador de protocolos teria o aspecto da figura 5.2 [PIAT80].

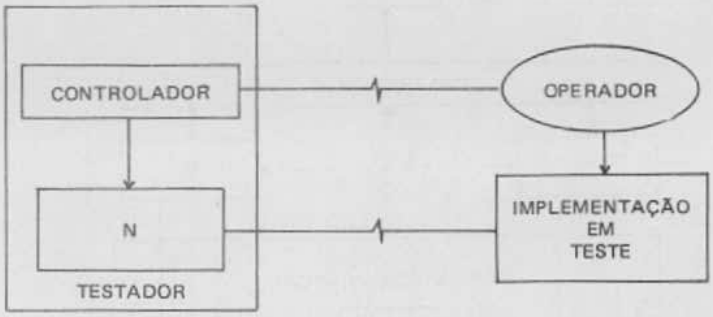

Figura 5.2. TESTADOR DE PROTOCOLOS

A idéia é basicamente a apresentada para o simulador. As diferenças básicas são: 1. o meio de comunicação agora é real (devendo ser o mais confiável possível, para que as mensagens enviadas sejam corretamente recebidas); 2. as entidades parceiras encontram-se agora em máquinas distintas - o controlador deve agora interagir com o operador do protocolo em teste e provavelmente não será possível conhecer o estado da entidade em teste, diretamente.

Em princípio o teste deveria tentar verificar todas as transições possíveis mas, como o tempo não permite, deveríamos dividi-las em duas categorias: as que *devem* ser tentadas e as que serão tentadas *aleatoriamente*, se o tempo permitir [PIAT80].

A implementação do protocolo no testador seria direta a partir de sua descrição formal. No entanto, para a geração das sequências de testes, é necessário que o projetista especifique quais são as essenciais e quais são as que têm pouca probalidade de ocorrer.

6. CONCLUSÕES

Este trabalho apresentou um método de descrição de protocolos – a Gramática de Transmissão – e a sua utilização dentro de um ciclo de vida completo do protocolo.

Foram apresentados dois dos sistemas já desenvolvidos com a finalidade de dar suporte ao projeto de protocolos e fornecer im

plementações do mesmo, automaticamente. A implementação original destes sistemas foi feita em Pascal, num PDP-11/34 rodando RSX-11M.

A seguir foram apresentadas duas outras ferramentas que pretende-se implementar no futuro: um simulador e um testador de protocolos.

Os trabalhos correlatos apresentados demonstram a viabilidade de uma abordagem deste tipo e reforçam o interesse em desenvolver ferramentas como as apresentadas, e outras, que facilitem o desenvolvimento, implementação e testes de protocolos.

7. BIBLIOGRAFIA

[BART69] - BARTLETT,K.A. et al. - A Note on Reliable Full-Duplex Transmission over Half-Duplex Lines. Communications of te ACM, 12(5): 260-1, May 1969.

[BART80] - BARTLETT,K. & RAYNER,D. - The Certification of Data Communication Protocols. NBS Trends and Applications Conference, p.12-7, May 1980.

[BLUM82] - BLUMER,T.P. & TENNEY,R.L. - A Formal Specification Technique and Implementation Method for Protocols. Computer Networks, 6(3): 201-17, July 1982.

[BOCH78] - BOCHMANN,G.V. - Finite State Description of Communication Protocols. Proceedings Computer Network Protocols Symposium, p. F 3/1-11, Liège, Belgium, Feb. 1978.

[BOCH80] - BOCHMANN,G.V. & SUNSHINE,C.A. - Formal Methods in Communication Protocol Design. IEEE Transactions on Communications, 28(4): 624-31, Apr. 1980.

[BRAN78] - BRAND,D. & JOYNER,JR,W.H. - Verification of Protocols Using Sympolic Execution, Computer Networks, 2:351-60, 1978.

[CHUN78] - CHUNG,P. & GAIMAN,B. - Use of State Diagrams to Engineer Communications Software.Proceedings of the Third International Conference on Software Engineering, p. 215-21, 1978.

[DANT78] - DANTHINE,A.& BREMER,T. - Modelling and Verification of end-to-and Transport Protocols. Computer Networks, 2(4/5): 381-95, Oct. 1978.

[DANT80] - DANTHINE,A.S. - Protocol Representation with Finite-Satate Models. IEEE Transactions on Communications, 28(4): 632-43, Apr. 1980.

[HAIL80] - HAILPERN,B.& OWICKI,S.- Verifying Network Protocols using Temporal Logic. IEEE Proceedings Trends & Applications, Computer Network Protocols, p.18-28, May,1980

[HARA78] - HARAGONZÓ,T. - Protocol Definition with Formal Grammars. Procedings Computer Network Protocols Symposium, p. F 6/1-10, Liège, Belgium, Feb. 1978.

[HOPC69] - HOPCROFT,J.E.& ULLMAN,J.D. - Formal Languages and their Relation to Automata. Addison-Wesley Publishing Co., Massachusetts, 1969.

[MENA81] - MENASCÉ,D.A., ROENICK,R.,FARRAN,L.Y.E. - Uma Linguagem de Especificação de Protocolos. Relatório Técnico, Deptº de Informática, PUC, Rio de Janeiro, 1981.

[MERL76] - MERLIN,P.M. - A Methodology for the Design and Implementation of Communication Protocols. IEEE Transactions on Communications, 24(6): 614-21, Jun. 1976.

[MERL79] - MERLIN,P.M. - Specification and Validation of Protocols. IEEE Transactions on Communications, 27(11): 1671-80, Nov. 1979.

[MONT82] - MONTEIRO,J.A.S. - Descrição, Validação e Geração Automática de Implementações de Protocolos. Tese de Mestrado apresentada à Escola Politécnica da USP, 1982.

[MONT83] - MONTEIRO,J.A.S. - Projeto e Geração Automática de Implementações de Protocolos in: V CONGRESSO REGIONAL DE INFORMÁTICA, Olinda, 1983. Anais. Recife, SUSESU-PE, 1983. P. 141-8.

[PIAT80] - PIATKOWSKI,T. - Remarks on ADCCP Validation and Testing Techniques. NBS Trends and Applications Symposium, p. 367-82, May 1980.

[SCHU80] - SCHULTZ,G.D. et al. - Executable Description and Validation of SNA. IEEE Transactions on Communications, 28(4): 661-77, Apr. 1980.

[SCHW81] - SCHWABE,D. - Formal Techniques for the Specification and Verification of Protocols. University of California at Los Angeles - Computer Science Department, Report nº CSD-810401, Apr. 1981.

[SMIT83] - SMITH,F.D. & WEST,C.H. - Technologies for Network Architecture and Implementation. IBM Journal of Research and Development, 27(1): 68-78, Jan. 1983.

[STEN76] - STENNING,N.V. - A Data Transfer Protocol. Computer Networks, 1(2): 99-110, Sept. 1976.

[SUNS79] - SUNSHINE,C. - Formal Techniques for Protocol Specification and Verification, Computer, 12(9):20-27,Sept.1979.

[TENG80a] - TENG,A.Y. - Protocol Constructions for Communication Networks. Ph.D. Dissertation, Department of Computer and Information Science, The Ohio State University, p. 256, Mar. 1980.

[TENG80b] - TENG,A.Y. - The Transmission Grammar Model for Protocol Construction. Proceedings Trends & Applications - Computer Network Protocols, p. 110-20, May 1980.

[WEIR78] - WEIR,D.F., et al. - X.25 Facilities on Detapac. Proceedings of the Fourth International Conference on Computer Communications, p. 273-9, Kyoto, Japan, Sept. 1978.

[WEST78] - WEST,C.H. - General Technique for Communications Protocol Validation. IBM Journal of Research and Development, 22(4): 393-404, Jul. 1978.

[ZIMM80] - ZIMMERMANN,H. - OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection. IEEE Transactions on Communications, 28(4): 425-32, Apr. 1980.