ESPECIFICAÇÃO FORMAL E IMPLEMENTAÇÃO DO PROTOCOLO DE TRANSPORTE

Otto Carlos M. B. Duarte Gilberto G. de Magalhães

Programa de Engenharia Elétrica
COPPE/EE - Universidade Federal do Rio de Janeiro
CP. 68504 - CEP 21945 - Rio de Janeiro, RJ
Tel: 280.8832 R. 413 - Telex (021) 33817 - UFCO BR

RESUMO

O desenvolvimento e a grande aceitação das redes de computadores foi um processo rápido que encontrou como principal obstáculo a dificuldade de interconexão de sistemas heterogêneos. Para transpô-lo, orgãos internacionais de padronização criaram o Modelo de Referência para a Interconexão de Sistemas Abertos.
O modelo propõe a repesentação de sistemas de maneira abstrata através de uma arquitetura em sete camadas, definindo o serviço oferecido por cada camada e especificando o protocolo para a interação de subsistemas remotos pertencentes a mesma camada. A camada Transporte é a quarta do modelo e oferece um serviço adequado de transferência de dados transparente fim-a-fim entre entidades da camada superior.
Um passo importante antes de partir para uma implementação coerente com o modelo de referência é a utilização das Técnicas de Descrição Formal (TDFs) que são linguagens capazes de definir o comportamento de sistemas de maneira completa, formal e sem ambiguidades tendo aplicabilidade na especificação de protocolos e serviços.
Este artigo mostra uma especificação formal do protocolo Transporte Classe 2 e a sua consequente implementação. O protocolo foi especificado utilizando a TDF Estelle. A implementação foi feita em PASCAL com o auxílio de um sistema operacional distribuído multitarefa.

Introdução

A dificuldade de interconexão de sistemas heterogêneos levou a "International Standards Organization" (ISO) à definir padrões através do modelo de referência denominado "Open Systems Interconnection" (OSI)[1]. Este modelo estabelece o conceito de sistemas abertos, isto é, sistemas que obedecem a determinados padrões e, portanto, são capazes de interagir com outros sistemas que seguem os mesmos padrões. Para conseguir esta compatibilidade, o modelo propõe uma arquitetura de sete camadas, sendo o sistema decomposto de maneira abstrata em um conjunto de subsistemas, cada um cor-

respondendo a uma camada. A camada mais baixa está relacionada com os aspectos físicos da commicação enquanto a últimm estabelece contato direto com o usuário. Cada camada é caracterizada pela definição do serviço oferecido à camada adjacente superior e por um protocolo que especifica as regras para a interação entre subsistemas da mesma camada, porem pertencentes a sistemas diferentes.

As definições dos serviços e as especificações dos protocolos estão escritas em normas, de maneira informal, podendo dar margens a diferentes interpretações. Além disso, cada uma delas contem detalhes específicos espalhados em seu interior que só não passam desapercebidos após sucessivas leituras. O esquecimento de um destes detalhes na hora de implementar pode por em risco a compatibilidade total entre sistemas.

Uma tentativa de descrição mais precisa encontra-se no anexo do protocolo da camada Transporte [2,3] e faz representações em máquinas de estados finitos. Entretanto, máquinas de estados são excelentes modelos para sistemas sequenciais. Dessa forma, o anexo caracteriza bem aspectos sequenciais do protocolo, porém não caracteriza aspectos de concorrência.

A necessidade de organizar a especificação informal em algo mais estruturado antes de se partir para uma implementação é fundamental. A solução está na utilização das chamadas Técnicas de Descrição Formal (TDFs)[4] que são linguagens capazes de definir o comportamento de sistemas de maneira completa, formal e precisa tendo aplicabilidade na especificação de protocolos e serviços do modelo de referência OSI. Tal como uma linguagem computacional, as TDFs definem regras de sintaxe e semântica que geram descrições sem ambiguidades.

O objetivo deste artigo é descrever uma especificação formal do protocolo de Transporte Classe 2 e sua consequente implementação. Primeiramente, a camada Transporte é descrita. Em seguida apresenta-se a linguagem Estelle, que foi a TDF escolhida para especificar este protocolo. Finalmente, aspec-

tos da implementação são abordados.

A Camada Transporte

A camada Transporte está localizada entre as camadas Rede e Sessão.As entidades de sessão podem exigir um serviço de transferência de dados confiável e econômico que esteja além da qualidade do serviço oferecido pela camada Rede. Cabe assim, ao Transporte, aprimorar o serviço recebido da Rede e oferece-lo à sessão.Esta passa ,então, a usufruir de um serviço de transferência de dados transparente fim-a-fim, entre entidades de sessão, cuja qualidade está de acordo com os seus desejos e necessidades.

O acesso ao serviço Transporte é feito atraves das primitivas de Transporte descritas na tabela 1.

Facilidades	Primitivas de Serviço
Abertura de Conexão	T-CONNECT request T-CONNECT indication T-CONNECT response T-CONNECT confirm
Transferência de Dados	T-DATA request T-DATA indication T-EXPEDITED-DATA request T-EXPEDITED-DATA indication
Fechamento de Conexão	T-DISCONNECT request T-DISCONNECT indication

tabela 1 - Primitivas do serviço Transporte.

Existem cinco classes de protocolo Transporte. Todas definem regras que disciplinam as fases de abertura, transferência de dados e fechamento de conexões de Transporte. A opção pelo uso de determinada classe é função da necessidade de melhoria do serviço Rede.

A classe 0 (Classe Simples) oferece o tipo mais simples de conexão de Transporte. O seu uso é aconselhável quando as conexões de Rede possuem aceitáveis taxas de erro residual e de erros sinalizados.

A classe 2 (Classe com Multiplexação) também é própria para quando se tem conexões de Rede com as mesmas características apresentadas para a classe 0. As diferenças estão no fato de que as conexões de Transporte podem ser multiplexadas em uma única conexão de Rede e existe a possibilidade de escolha pelo uso de controle de fluxo explícito.

A classe 1 (Classe com Recuperação de Erros Básicos) cria conexões de Transporte tolerantes a possíveis desconexões de Rede e, por isso são próprias quando as conexões de Rede possuem taxa de erro residual aceitável, porém deficiente taxa de erros sinalizados.

A classe 3 (Classe com Multiplexação e Recuperação de Erros) reune as características das classes 1 e 2.

Por último existe a classe 4 (Classe com Detecção e Recuperação de Erros), cujas conexões de Transporte possuem as características da classe 3 adicionada da capacidade de detecção e recuperação de erros provocados por um serviço Rede pouco confiável.

O diálogo entre entidades de Transporte pertencentes a sistemas distintos é feito através da troca de Unidades de Dados de Protocolo de Transporte, ou TPDUs ("Transport Protocol Data Units").Os TPDUs são compostos por octetos e podem ser dos seguintes tipos :

. CR ("Connection Request") - Pedido de abertura de conexão

. CC ("Connection Confirm") - Confirmação de abertura de conexão

. DT ("Data") - Envio de dados

. ED ("Expedited Data") - Envio de dados urgentes

. AK ("Data Acknowledgment") - Reconhecimento de dados

. EA ("Expedited Data Acknowledgment") - Reconhecimento de dados urgentes

. DR ("Disconnect Request") - Pedido de fechamento de conexão

. DC ("Disconnect Confirm") - Confirmação de fechamento de conexão

. RJ ("Reject") - Rejeição

. ER ("TPDU Error") - Erro em TPDU

Os dados das entidades de sessão são enviados à camada Transporte em Unidades de Dados do Serviço Transporte, ou TSDUs ("Transport Service Data Units"). Os TSDUs são, então, segmentados em TPDUs DT sendo estes mapeados em Unidades de Dados do Serviço Rede (NSDUs) e enviados para a camada inferior.

A Linguagem Estelle

Entre as técnicas formais existentes está a linguagem Estelle [5] padronizada pela ISO que é uma TDF em geral usada para sistemas de processamento concorrente e distribuído.

Uma especificação em Estelle descreve um conjunto de módulos, visualizados sob os pontos de vista interno e externo, que podem trocar mensagens (interações) entre si através de ligações bidirecionais (canais) de suas respectivas portas (pontos de interação). Assim, *módulos*, *canais*, *interações* e *pontos de interação* são elementos declarados em uma especificação em Estelle e através deles fica representado o dinamismo de sistemas pelas possibilidades de :

- criação e destruição de módulos;

- ligação ou desligamento de pontos de interação;
- envio e recepção de interações através dos pontos de interação que possuem uma fila tipo FIFO ("First-In-First-Out").

O comportamento interno de um módulo é caracterizado como sendo um sistema de transição de estados não determinístico. Em outras palavras, o modulo é representado internamente como uma máquina que possui :

- um conjunto de estados;
- um subconjunto de estados iniciais;
- uma relação próximo estado.

As ações internas ocorrem devido ao disparo de transições. Sintaticamente elas são compostas por duas partes : cláusulas e bloco da transição. As cláusulas determinam, para um estado, as condições que deverão estar satisfeitas para que a transição esteja habilitada e o novo estado caso ela venha a disparar. O seu disparo dependerá da hierarquia e dos atributos dos modulos. Disparar uma transição significa executar a ação que está definida no bloco da transição. Este bloco é escrito em linguagem PASCAL ISO com algumas restrições. A execução de uma transição é uma operação atômica. Por estas caracteristicas pode-se dizer que a linguagem Estelle é uma TDF baseada em um modelo de máquinas de estados finitos estendidos.

A seguir estão descritas algumas caracteristicas do módulo.

O Módulo

O módulo está sempre associado a duas definições : o cabeçalho, que caracteriza a visibilidade externa, e o corpo, que descreve o seu comporta-

mento interno. Várias instâncias de um módulo podem ser criadas e para diferenciá-las são utilizadas váriáveis representantes de instâncias. A declaração destas variáveis está na tabela 2 juntamente com a descrição dos principais elementos de Estelle. A criação é feita através da primitiva INIT (tabela 3).

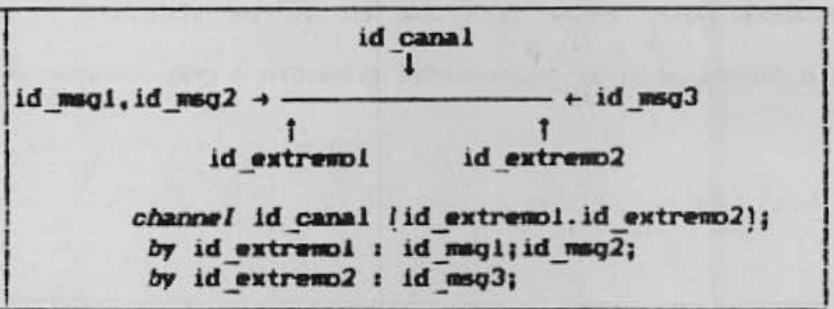
canal	 *channel* id_canal (id_extremo1,id_extremo2); *by* id_extremo1 : id_msg1;id_msg2; *by* id_extremo2 : id_msg3;
ponto de interação	*ip* id_ponto_int : id_canal (id_extremo) *individual queue*;
cabeçalho	*module* id_cabecalho atributo (P1:P1_type; ...); *ip* . {declaração dos pontos de } . {interação internos } *export* . {declaração de variáveis } . {exportadas } *end*;
variável de instância	*modvar* id_var : id_cabecalho;
estados	*state* id_st1, ... ,id_stn;
transição	*trans* *when* id pont_int.msg1 {chegada de mensagem} *from* st1 {estado inicial } *provided* P1 {condição booleana } *priority* n {prioridade } *to* st2 {próximo estado } *begin* . {bloco da transição } . *end*;

tabela 2 - Sintaxe de declaração dos elementos principais de Estelle

O corpo é dividido em três partes : declaração; inicialização e declaração das transições. A declaração pode conter definições de constantes, tipos, canais, cabeçalhos, corpos, pontos de interação internos, variáveis comuns e representantes de instâncias, posíveis estados, procedimentos e funções.

Módulos podem estar contidos em outros módulos. O módulo interior recebe a denominação de descendente enquanto o que o contém é o ascendente.

INIT	cria e inicializa	*init* id_inst *with* id_corpo
RELEASE	destroi instância de módulo	*release* id_inst
CONNECT / DISCONNECT	liga/desliga pontos de interação	*connect* id_pont_int1 *to* id_pont_int2 *disconnect* id_pont_int1 *to* id_pont_int2
ATTACH / DETACH	liga/desliga pontos de interação externos de módulos ascendentes e descendentes	*attach* id_pont_int1 *to* id_pont_int2 *detach* id_pont_int1 *to* id_pont_int2
OUTPUT	envio de interações através de pontos de interação	*output* id_pont_int.msgi

tabela 3 - Primitivas de Estelle

O poder do módulo ascendente sob os seus descendentes é evidenciado por:

- prioridade de execução, uma vez que as suas transições tem prioridade

de disparo;

- possibilidade de criar, destruir e alterar a configuração de conexões de seus descendentes;
- acesso à algumas variáveis do descendente (variáveis exportadas).

Além da hierarquia ascendente/descendente, a execução depende também do atributo do módulo podendo ser paralela (atributo *process*), sequencial (*activity*) ou totalmente aleatória (*systemprocess* ou *systemactivity*).

Aplicação de Estelle na Especificação do Transporte

A figura 1 mostra a estrutura hierárquica dos módulos de uma especificação em Estelle do protocolo Transporte Classe 2. As camadas superiores e inferiores à Transporte estão representadas, respectivamente, pelos módulos Camadas_Sup e Camadas_Inf. Os corpos destes módulos não fazem parte desta especificação e, por isso são declarados como externos. No anexo encontra-se de maneira resumida algumas partes da especificação. O que foi omitido é muito semelhante a linguagem PASCAL.

As interações trocadas entre módulos são primitivas de serviço. A abertura e fechamento de conexões de Transporte estão relacionados, respectivamente, com a criação e destruição de módulos Conexão, feitos pelo módulo Transporte.

A máquina de estados do anexo da norma Transporte é praticamente toda representada pelas transições contidas no corpo dos módulos Conexão. Cabe ao módulo Transporte a execução dos procedimentos iniciais de abertura de conexão até a criação do módulo Conexão correspondente.

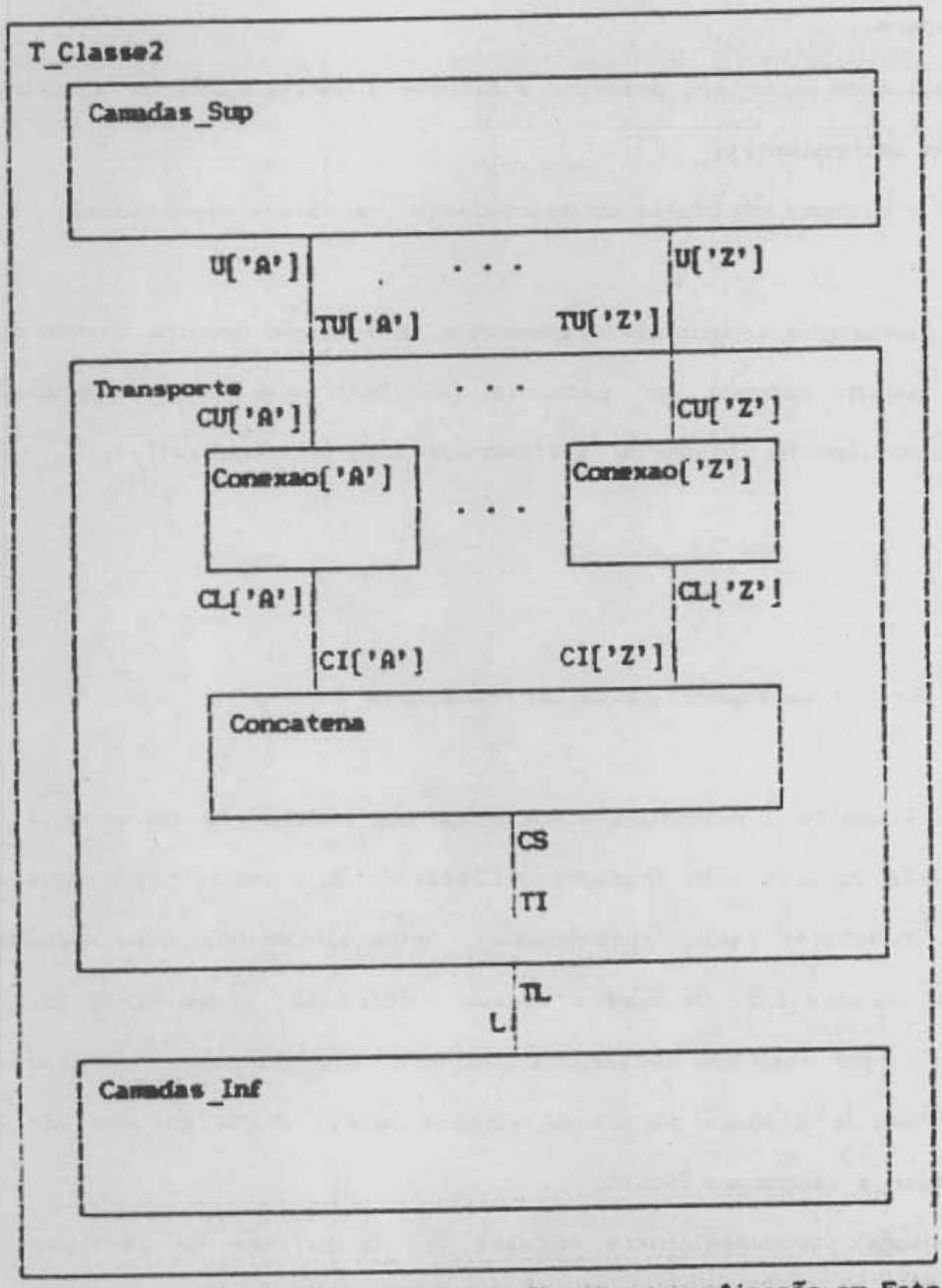


figura 1 - Estrutura hierarquica de uma especificação em Estelle para o protocolo de Transporte Classe 2.

Implementação do Protocolo Transporte

A implementação do protocolo foi feita a partir da especificação em

Estelle descrita anteriormente. A concorrência entre os módulos e o fato de Estelle ser baseada em PASCAL são dois aspectos marcantes que levam ao uso, na implementação, da linguagem PASCAL e de um ambiente operacional multitarefa. O estreito relacionamento de uma camada com as adjacentes exige um oferecimento do serviço Rede e requisições do serviço Transporte por parte da camada Sessão.

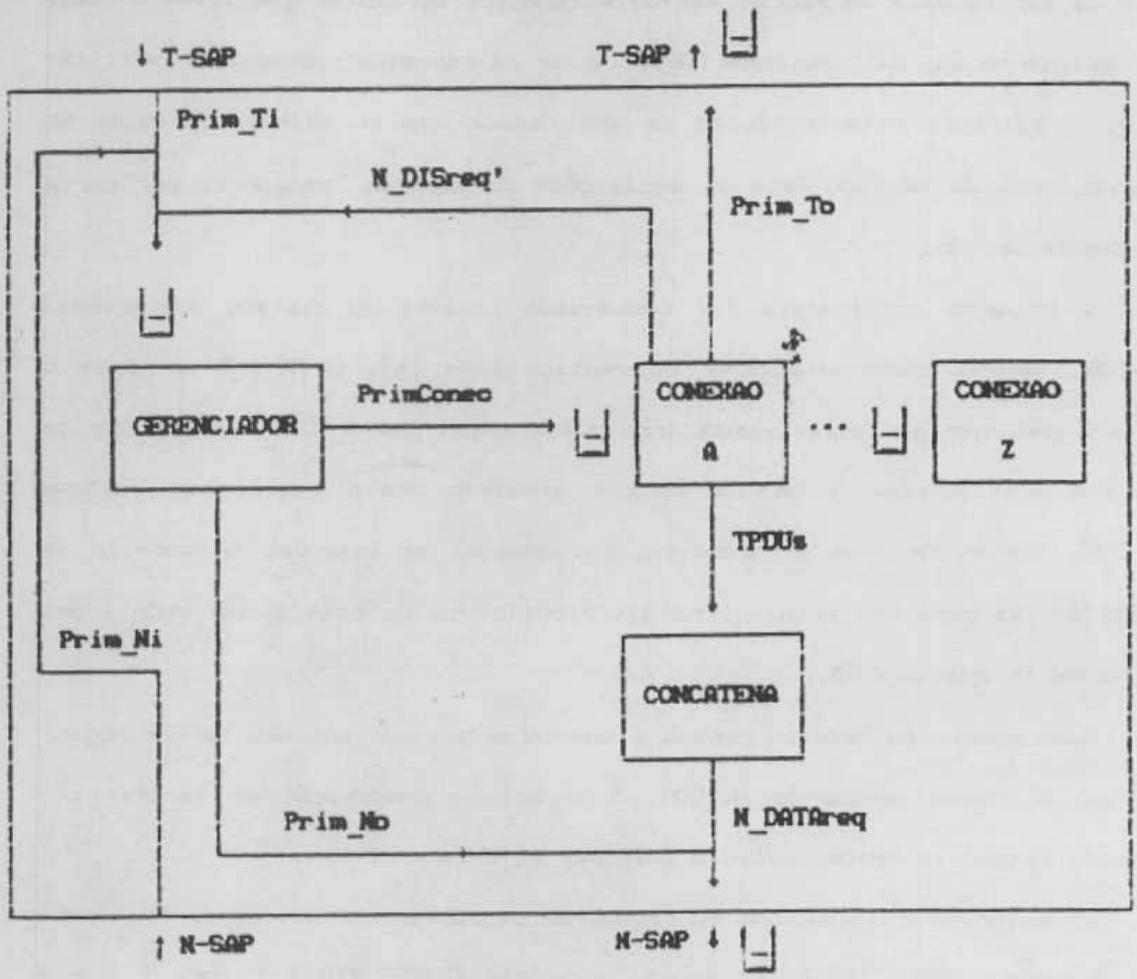
O ambiente multitarefa foi conseguido através do sistema operacional MT_DOS, desenvolvido pela Mira Informática Ltda [6]. O MT_DOS converte o MS_DOS presente em microcomputadores compatíveis com o IBM PC/XT/AT em um sistema multitarefa. O serviço Rede é simulado com o auxílio do sistema MT_DOS, também da Mira Informática, responsável em estender o conceito de multitarefas para uma arquitetura distribuída que no caso é uma rede local da Cetus Informática SA.

Cada módulo em Estelle passou a ser caracterizado por uma tarefa rodando sob o gerenciamento do MT_DOS. A criação e destruição de tarefas são feitos, respectivamente, pelas primitivas *mt_cria* e *mt_fim*.

A figura 2 é o diagrama em blocos de um subsistema da camada Transporte. A tarefa GERENCIADOR implementa o módulo TRANSPORTE (figura 1) e é criada logo no início da execução do programa. Para cada conexão de Transporte aberta, a tarefa GERENCIADOR cria uma tarefa CONEXAO. Assim, o número de conexões abertas corresponde ao número de tarefas CONEXAO(A..Z) que rodam no sistema executando o mesmo código. A hierarquia entre módulos faz com que as tarefas CONEXAO tenham prioridade inferior ao GERENCIADOR na disputa pelo uso do processador.

Um detalhe interessante diz respeito à implementação das filas dos pontos de interação. Elas são recursos compartilhados por tarefas que as acessam para a colocação e retirada dos elementos descritos na figura 2, e consequentemente, devem ser tratadas como regiões críticas. O ideal seria

dispor de um sistema operacional possuidor de um gerenciador de filas que



Prim_Ti = [T-CONreq, T-CONresp, T-DATAreq, T-EDreq, T-DISreq]

Prim_Ni = [N-CONind, N-CONconf, N-DATAind, N-DISind, N-RSTind]

PrimConec = [N-CONconf, N-DISind, N-RSTind, TPDU, T-EDreq, T-DISreq, T-CONresp, T-DATAreq]

Prim_To = [T-CONind, T-CONconf, T-DATAind, T-EDind, T-DISind]

Prim_No = [N-CONreq, N-CONresp, N-DISreq, N-RSTresp]

figura 2 - Diagrama em blocos da implementação do Protocolo Transporte

tornaria o tratamento destas transparente ao implementador. Um exemplo seria

a utilização de um esquema de envio e recepção de mensagens assíncrono. O sistema operacional se encarregaria de guardar as primitivas enviadas para posteriormente remetê-las em ordem quando a tarefa recebedora pudesse tratá-las. Como no MT_DOS o número de mensagens pendentes é limitado, as filas foram criadas com variáveis dinâmicas e o problema de sincronizar o acesso foi resolvido pela utilização de semáforos.

O MT_DOS possui primitivas para a alocação, liberação de semáforos (*mt_proximo_smf* e *mt_libera*) e execução das operações "P" e "V" (*mt_p* e *mt_v*). Existem três semáforos associados a cada fila. Um é binário com valor inicial unitário cuja função é garantir a exclusão mútua. O início do acesso à fila é marcado por uma operação P sobre este semáforo e o seu término por uma operação V. Os outros dois controlam o número de posições vazias e cheias para que não haja problema de estouro de fila.Em resumo, são necessárias as seguintes operações :

Colocação em filas

```
P(SmfVazio);
P(SmfMutex);
    :
V(SmfMutex);
V(SmfCheio);
```

Retirada das filas :

```
P(SmfCheio);
P(SmfMutex);
    :
V(SmfMutex);
V(SmfVazio);
```

Todos os aspectos intrínsecos de Estelle (modularidade, hierarquia,...)

são respeitados na implementação,pois esta foi resultante de um mapeamento direto da especificação. Vale ressaltar, porém, duas exceções que se referem a destruição de módulos/tarefas e à atomicidade das transições. A primeira é evidenciada pelo fato de que todos os módulos foram mapeados em tarefas. O sistema MT_DOS só permite que a própria tarefa se destrua através da execução da primitiva *mt_fim* enquanto Estelle não permite que um módulo se autodestrua. A atomicidade não foi totalmente respeitada, pois a sua implementação através de semáforos resultaria na execução de muitas operações P e V para garantir, por exemplo, que o bloco de uma transição não viesse a ser interrompido devido a ativação de uma tarefa prioritária. Haveria uma solução fácil e eficaz caso o sistema operacional possuisse um esquema de prioridade dinâmica para tarefas.

No estágio atual, pedidos do serviço Transporte são feitos por tarefas capazes de enviar e receber primitivas de Transporte. A qualquer instante pode-se requisitar, via teclado, a formação e o envio de primitivas. A recepção é constatada por mensagens no terminal. Esta facilidade ajudou o teste e a análise do comportamento do protocolo implementado. O próximo passo será a automatização dos pedidos e respostas para posterior analise de desempenho.

Conclusão

A solução encontrada neste trabalho foi fruto de um mapeamento direto da especificação em linguagem Estelle. O procedimento foi feito para os protocolos de Transporte Classes 0 e 2, sendo os resultados incentivadores ao desenvolvimento de ferramentas automáticas de auxílio à implementação de

protocolos de comunicação.

A estrutura em módulos permitiu a utilização de uma abordagem de "cima-para-baixo", tal qual o processo de desenvolvimento de sistemas, evoluindo de um nível amplo e conceitual para um nível crescentemente detalhado e operacional.

O grande ganho foi uma solução estruturada que possui facilidades de :

- execução de testes;
- inclusão ou exclusão dos serviços opcionais oferecidos pela norma;
- adaptação do protocolo a diferentes interfaces entre camadas;
- compreensão da implementação até mesmo por aqueles que não participaram diretamente.

A escolha de um sistema operacional adequado é de extrema importância. O oferecimento de um gerenciamento de filas faz com que o tratamento destas seja transparente ao implementador. É desejável também a presença de um esquema de prioridade dinâmica de tarefas para implementar atomicidade. O uso apenas de recursos básicos do MT_DOS torna fácil a adaptabilidade a qualquer outro ambiente operacional.

Referências Bibliográficas

[1] ISO 7498, "Basic Reference Model for Open Systems Interconnection", ISO, 1983;

[2] ISO 8072, "Information Processing Systems - Open Systems Interconnection - Transport Service Definition", ISO, 1983;

[3] ISO 8073, "Information Processing Systems - Open Systems Interconnection - Connection Oriented Transport Protocol Especification", ISO, 1984;

[4] DICKSON G. J. e CHAZAL P. E., "Status of CCITT Description Techniques and Application to Protocol Specification", Proceeding of the IEEE, vol. 71 pp. 1346 - 1355, Dec. 1983;

[5] ISO/TC 97/SC 21/WG1/FDT/B, "ESTELLE - A Formal Description Techniques Based on an Extended Transition Model", ISO, 1986;

[6] MOTTA J. e RAMOS J., "Sistema Mira para Redes Locais",Anais do Primeiro Congresso Nacional da Tecnologia do Software, Telemática e Informação" - RJ, 1987;

[7] "Sistema Mira para Redes Locais", Manual de Operação,Mira Informática RJ, 1988;
[8] MOURA J. A. B., SAUVE J. P., GIOZZA W. F. e ARAUJO J. M., "Redes Locais de Computadores - Protocolos de Alto Nível e Avaliação de Desempenho", McGraw-Hill, São Paulo, 1986;
[9] CHONG H. Y., "Software Development and Implementation of NBS Class-4 Transport Protocol",Computer Networks and ISDN Systems, num 11 pp. 353 - 365,Nov. 1986;
[10] PETERSON J. L. e SILBERSCHATZ A., "Operating Systems Concepts", ADDISON-WESLEY, 1983;

Anexo - Especificação em Estelle do protocolo Transporte

Transporte Classe 2

```
specification T_Classe2 systemprocess ;
{começo das declarações da especificação}

const
  Tmin = any integer;
  Tmax = any integer;
     .
     .              {definição de constantes}
     .
type
  T_SAPtp = Tmin .. Tmax;
  N_SAPtp = Nmin .. Nmax;
     .
     .               {definição de tipos}

channel AcessoSup(Cima,Baixo);     {definição dos canais}
  by Cima :
    T_CONreq(...);
    T_CONresp(...);
    T_DATAreq(...);
    T_EDreq(...);
    T_DISreq(...);
  by Baixo :
    T_CONind(...);
    T_CONconf(...);
    T_DATAind(...);
    T_EDind(...);
    T_DISind(...);

channel AcessoInf(Cima,Baixo);
  by Cima :
    N_CONreq(...);
    N_CONresp(...);
    N_DATAreq(...);
    N_DISreq(...);
    N_RSTresp(...);
  by Baixo :
```

```
    N_CONind(...);
    N_CONconf(...);
    N_DATAind(...);
    N_DISind(...);
    N_RSTind(...);

module Camadas_Suptp process (...);     {definições de cabeçalhos}
  ip  U: array['A' .. 'Z'] of AcessoSup(Cima) common queue;
end;              {fim do cabeçalho Camadas_Suptp}

module Transportetp process (...);
  ip  TU: array['A' ..'Z'] of AcessoSup(Baixo);
      TL: AcessoInf(Cima);
end;              {fim do cabeçalho Transportetp};

module Camadas_Inftp process (...);
  ip  L: AcessoInf(Baixo);
end;

body Camadas_Supby for Camadas_Suptp; external     {definição dos corpos}
body Camadas_Infby for Camadas_Inftp; external
body Transporteby for Transportetp;

          {está definido a seguir}

end;      {fim do corpo Transporteby}

modvar    {declaração de variáveis de instâncias}
  Camada_Sup: array[T_SAPtp] of Camadas_Suptp;
  Transporte: array[T_SAPtp] of Transportetp;
  Camadas_Inf: Camadas_Inftp;

var       {declaração de variáveis}
  S: T_SAPtp;
  X: char;
  N_SAP: N_SAPtp;


function ExtraiEndr(T_End:T_SAPtp):N_SAPtp;
primitive;          {"primitive" indica que a função depende}
                    {da implementação e por isso não está   }
                    {descrita.ExtraiEndR calcula o endereço }
                    {do N_SAP a partir do endereço do T_SAP }

                     {declaração de procedimentos e funções}

initialize           {inicialização da especificação}
begin
  init Camadas_Inf with Camadas_Infby;
  All S: T_SAPtp do
  begin
    N_SAP := ExtraiEndR(S);
    init Camadas_Sup[S] with Camadas_Supby (S);
    init Transporte[S] with Transporteby (S);
    All X: 'A' ..'Z' do
    begin
      connect Camadas_Sup[S].U[X] to Transporte[S].TU[X];
      connect Transporte[S].TL to Camadas_Inf.L[N_SAP];
```

```
    end;
  end;
end;
end.          {fim da especificação}
```

Corpo Transporte

```
body Transporteby for Transportetp;
      .
      .
      .
channel PaiFilho(Cima,Baixo);
  by Cima:
      .
      .
      .
channel AcessoInt(Cima,Baixo);
      .
      .
      .
module Conexaotp process (...);
ip CU: array['A' .. 'Z'] of PaiFilho(Baixo);
   CL: array['A' ..'Z'] of AcessoInt(Cima);
export
      .
      .
end;

body Conexaoby for Conexaotp;
      .
      .
end;
      .
      .
modvar
  Conexao: array['A' ..'Z'] of Conexaotp;
state WFCC,WBCL,OPEN,CLOSING,WFTRESP,CLOSED;
      .
      .
trans
  from WFTRESP
   to OPEN
    when CU.T_CONresp
     EnviaCC
      .
      .
end;          {fim do corpo Tranporteby}
```