Implementação de protocolo de comunicação em uma rede com integração de serviços utilizando a linguagem CHILL *

M. Cristina E. Ussami e Shusaburo Motoyama
Depto. Telemática, FEE - UNICAMP
Caixa Postal 6101 - CEP:13081 - Campinas, SP, Brasil
Email: Cris@ccvax.unicamp.br, Motoyama@ccvax.unicamp.br

Sumário

Neste trabalho são discutidos os aspectos de implementação de protocolo de comunicação em uma rede com integração de serviços (voz e dados) utilizando a linguagem CHILL. A rede com integração de serviços que está em desenvolvimento no Dep. de Telemática, da UNICAMP denominada RALFO, utiliza a fibra óptica como o meio de transmissão e é conveniente para aplicações em redes locais assim como em redes metropolitanas. Inicialmente são apresentados a configuração da rede, o esquema de acesso, a arquitetura em camada da rede e a estrutura de um nó da rede. A seguir são discutidos os aspectos de gerenciamento de um nó da rede e a escolha da linguagem de programação. São detalhados também a implementação do protocolo enlace de dados utilizando a linguagem CHILL e os testes de funcionamento do protocolo.

1 Introdução

A grande disseminação das redes locais de computadores originou a necessidade de interligação dessas redes por meio de uma outra rede local de velocidade de transmissão maior (velocidade média). Estas redes de velocidade maior que estão ainda no âmbito de uma área geograficamente fechada (como campus universitário ou indústrias de grande porte), por sua vez ocasionaram a necessidade de suas interconexões através de uma rede de alta velocidade, abrangendo agora áreas de dimensões metropolitanas, para conexões de equipamentos em locais bastante distantes. Essas redes, denominadas redes de áreas metropolitanas (MAN - Metropolitan Area Network), estão sendo objetos de padronização através do padrão IEEE 802.6 e do padrão FDDI do ANSI (American National Standard Institute).

*Este trabalho foi parcialmente financiado pela FAPESP, CNPq e Telebrás

Essas redes metropolitanas, além de transportar dados em geral, estão sendo preparadas também para transportar sinais de voz, para aumentar a eficiência da rede. Uma outra característica bastante importante nessas redes é que elas estão sendo desenvolvidas para ter máxima compatibilização com a RDSI-FL (Rede Digital de Serviços Integrados de Faixa Larga). A RDSI-FL está sendo considerada como uma rede que fará a integração definitiva de todas as redes existentes como as redes telefônicas, as LANs, as MANs e todas outras redes atualmente em desenvolvimento.

Neste trabalho são discutidos os aspectos de implementação de uma rede com integração de serviço que pode ser utilizada tanto em rede local como em rede metropolitana. A rede discutida não segue as padronizações acima citadas (IEEE 802.6 ou FDDI), mas é bastante flexível para se compatibilizar com a emergente RDSI-FL, em concordância com a tendência internacional.

A rede proposta utiliza a fibra óptica como o meio de transmissão e pode transportar dois tipos de sinais: sinais de voz e sinais de dados em geral (incluindo aqui dados de sinalização telefônica) e foi denominada RALFO (Rede de Area Local com Fibras Ópticas). O esquema de acesso ao meio é baseado em "Empty Slot", utilizado no anel de Cambridge, da Inglaterra, porém modificado para suportar o tráfego de voz [Guardi 4]. A arquitetura geral da rede é baseada em IEEE 802 com modificações para incorporar serviços telefônicos e foi discutida em [Carnei 5]. Os aspectos de implementação da interface de voz para a rede proposta foi detalhada em [Pessoa 3].

Os objetivos deste trabalho são discutir os aspectos de gerenciamento de um nó da rede proposta e apresentar a implementação do protocolo de comunicação da camada enlace de dados utilizando a linguagem CHILL.

Na seção 2 são apresentados resumidamente a configuração da rede, o método do controle de acesso ao meio, a arquitetura em camadas da RALFO e a estrutura de um nó da rede. Os aspectos gerais de gerenciamento de protocolos (sistema operacional) assim como a escolha da linguagem de programação de protocolos estão discutidos na seção 3. Na seção 4 é detalhada a implementação do protocolo utilizando a linguagem CHILL e é mostrado também o teste de funcionamento do protocolo. Finalmente na seção 5 são apresentadas as principais conclusões do trabalho.

2 Arquitetura da RALFO

2.1 Configuração

A RALFO apresenta topologia em anel duplo, tendo fibras óticas como meio de transmissão. A configuração em anel duplo aumenta o grau de tolerância a falhas da rede. No caso de falha em um dos enlaces, faz-se uma reconfiguração da rede através de um caminho alternativo ("loop back") entre os nós onde o enlace se encontra danificado, e para o caso de falha em um dos nós, aplica-se a técnica de isolação ("by pass") do nó.

A configuração em anel permite a rede operar tanto em distâncias curtas (rede local) quanto em distâncias alcançando regiões metropolitanas (rede metropolitana),

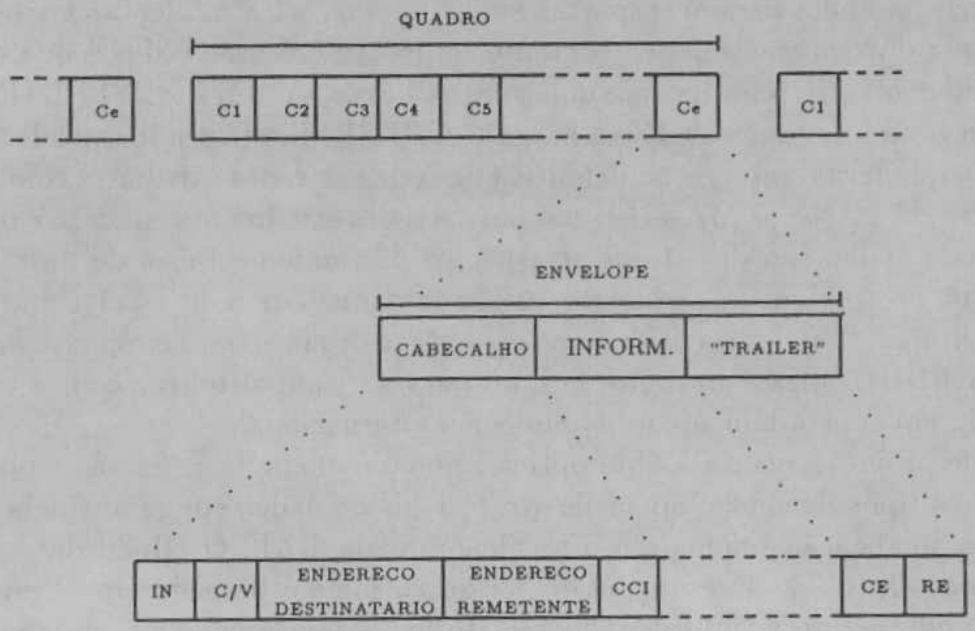

Figura 1: Estrutura de quadros e envelopes

pela introdução de regeneradores em cada nó da rede.

2.2 Método de acesso e estrutura de quadros

Foi adotada na RALFO, a comutação de pacote tanto para voz como para dados; entretanto os pacotes de voz têm maior prioridade no acesso ao meio. O método de acesso adotado foi o "Empty Slot", modificado para incluir os sinais de voz [Guardi 4]. A estrutura de quadros está representada na figura 1.

Quando um nó da rede deseja transmitir dados ele deve aguardar por um canal vazio. Assim que encontrar, ele sinaliza o canal como cheio e transmite seus dados ao canal. O nó pode desprezar o canal caso ele tenha sido o seu usuário no quadro anterior, evitando o monopólio do mesmo.

Para sinais de voz digitalizados, o método foi modificado para evitar atrasos consideráveis. Como os sinais de voz aparecem em surtos onde a continuidade deve ser mantida, neste esquema de acesso proposto um canal é alocado para um surto e permanece alocado até que ocorra um novo intervalo de silêncio.

Para a recepção de dados, os nós ficam analisando continuamente os campos destinatários dos canais. Uma vez detectado seu próprio endereço, o nó faz uma cópia do conteúdo do canal. Em seguida o canal é retransmitido, porém com uma modificação indicando que o dado foi lido. O nó que enviou o dado é responsável por liberar o canal utilizado.

Este esquema de acesso proposto é muito conveniente, pois podemos compatibilizar com a RDSI-FL, colocando as células da RDSI-FL no campo de informação dos envelopes da RALFO.

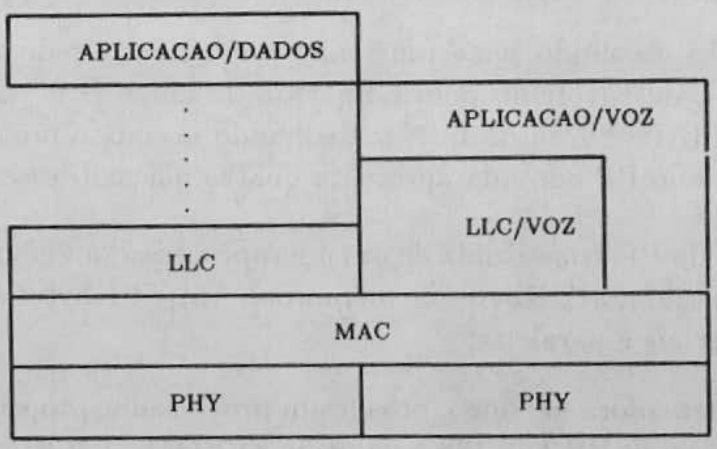

Figura 2: Arquitetura de camadas da RALFO

Este esquema de acesso proposto é muito conveniente, pois podemos compatibilizar com a RDSI-FL, colocando as células da RDSI-FL no campo de informação dos envelopes da RALFO.

2.3 Arquitetura de camadas da RALFO

A estrutura em camadas projetada para a RALFO está representada na figura 2.

- Camada PHY: responsável por ativar, manter e desativar conexões físicas para a transmissão de bits entre dois nós. Trata da conversão de pulsos elétricos para sinais ópticos e o procedimento inverso no recebimento de sinais ópticos.
- Camada MAC: tem como função fazer a interface entre as camadas LLC, LLC/Voz e a camada física, provendo o controle de acesso ao meio físico que na RALFO corresponde ao duplo anel de fibra óptica.
- Camada LLC: reponsável pelo controle de enlace lógico de dados e está implementada segundo as recomendações do padrão IEEE 802.2. Ela dará suporte às camadas superiores do modelo RM-OSI/ISO com objetivo de oferecer as facilidades de comunicação de dados.
- Camada LLC/Voz: responsável pelo controle de enlace de voz, cujo tratamento de sinalização tornou necessária a definição de um a camada apropriada.
- Camada Aplicação/Dados: Esta camada implementa uma aplicação de transferência de dados com o objetivo de testar a operacionalidade da rede. Ela comunica-se diretamente com a camada LLC, sem intermédio das demais camadas do modelo RM OSI/ISO (REDE, TRANSPORTE, SESSAO, APRESENTAÇÃO), para simplificar o protótipo a ser implementado.

2.4 Estrutura de um nó da RALFO

O equipamento escolhido para funcionar como nó da rede foi o Processador Preferencial (PP), desenvolvido pelo CPqD da Telebrás [PP 1]. O PP utiliza o microprocessador IAPx 80286 da Intel trabalhando no modo protegido.

A configuração do PP adotada apresenta quatro placas [Pessoa 3]:

- UPN : CPU do PP, constituída de um microprocessador 286 com co-processador matemático (287), 512 Kbytes de memória RAM, 64 Kbytes de EPROM e duas interfaces seriais e paralelas;
- COM : Controladora de vídeo, possui um processador próprio (8088), controla 3 interfaces seriais RS 232, 128 Kbytes de EPROM , 192 Kbytes de RAM e um bloco de memória Dual-Port . Através destas memórias é possível um outro processador escrever internamente na placa COM;
- DIS : Controladora de disco, com capacidade de controlar quatro unidades de disco rígido (tecnologia winchester) de 5,25 polegadas com capacidade formatada até 288 Mbytes cada, quatro unidades de disco flexível de 8 e 5,25 polegadas e duas unidades de fita cartucho;
- RAS : Placa rastreadora, de grande utilidade no desenvolvimento de projetos, pois através dela é posssível acompanhar o fluxo de informação no barramento do PP, simulando um analizador lógico. Apresenta um processador (8085) com até 32 Kbytes de EPROM e 8 Kbytes de RAM.

A estas placas serão acopladas a placa MAC e a Interface de Voz, através das quais faz-se o acesso à rede e aos terminais telefônicos respectivamente. Já os terminais de dados serão conectados através das linhas seriais existentes nas placas do PP. Vale ressaltar que tanto a interface de voz quanto a MAC possuem seus próprios processadores, tendo-se assim uma estrutura de multiprocessadores no nó da rede. A arquitetura do nó pode ser representada pela figura 3.

A distribuição das camadas, vistas no ítem 2.3, nas placas do PP está representada na figura 4 : a camada LLC assim como as camadas superiores de dados estão residentes na placa UPN ; a sinalização juntamente com o software de controle da interface de voz ficam na placa de voz e na MAC localiza-se o protocolo de acesso ao meio.

3 Aspectos gerais relacionados à implementação de protocolos

Na implementação de aplicações de comunicações de dados (transferência de arquivos, correio eletrônico), em um determinado computador, geralmente pensa-se em adicionar às funções normais do seu sistema operacional, funções especializadas no tratamento de comunicação.

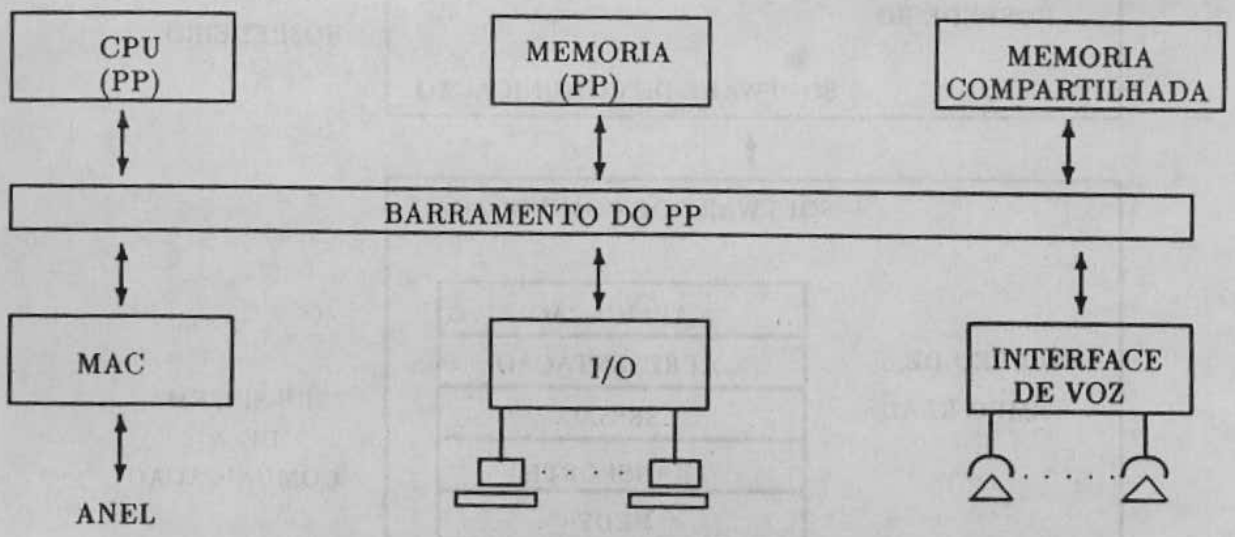

Figura 3: Arquitetura de hardware do PP

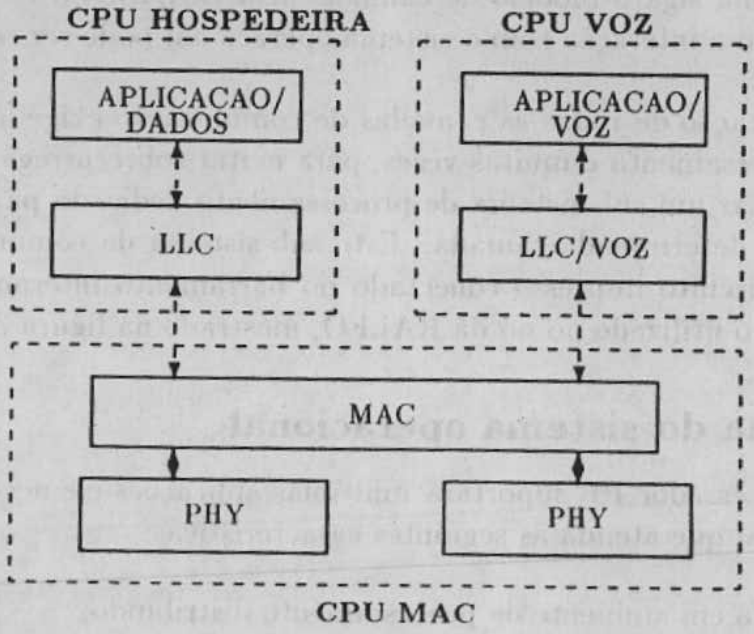

Figura 4: Distribuição das camadas nos processadores do nó

Figura 5: Interação das funções de comunicação - modelo OSI/ISO

Caso o sistema siga o modelo de camadas RM-OSI/ISO, a localização das camadas bem como a interação com o sistema operacional pode ser representada pela figura 5.

A implementação de todas as camadas de comunicação exige um esforço considerável de processamento e muitas vezes, para evitar sobrecarregar o computador, procura-se utilizar um sub-sistema de processamento dedicado para tratar os protocolos de uma determinada camada. Este sub-sistema de comunicação pode ser um cartão de circuito impresso conectado no barramento interno do processador hospedeiro, como utilizado no nó da RALFO, mostrado na figura 3.

3.1 Escolha do sistema operacional

Como o processador PP suportará múltiplas aplicações ele necessita de um sistema operacional que atenda às seguintes características:

- Multitarefa em ambiente de processamento distribuído;
- Núcleo em tempo real;
- Permitir escalonamento de processos baseados em prioridades;
- Possuir mecanismos de comunicação entre processos;

- Permitir o tratamento de interrupções;
- Ter mecanismos para gerenciamento de temporizadores.

O SOP, sistema operacional desenvolvido no CPqD da Telebrás pelo grupo de projeto PP-SO/P (Sistema Operacional para o PP operando no modo protegido) [SOP 2], satisfaz os requisitos acima e é próprio para utilização no PP.

O SOP é um sistema operacional configurável, isto é, pode-se decidir sobre a presença ou não de determinados módulos executáveis. Além disso, ele opera com o conceito de processo (usando o recurso "tarefa" do iAPX 286).

Alguns outros sistemas operacionais foram analisados e constatados deficientes nos aspectos relacionados a controle de processo em tempo real. Geralmente o escalonamento era baseado na divisão do tempo.

3.2 Escolha da linguagem de programação

A linguagem de programação deve atender aos seguintes requisitos:

- Geração de código executável para o microprocessador Intel 286;
- Suporte a execução concorrente de processos;
- Suporte a comunicação entre processos;
- Modularidade na programação.

Uma vez escolhida a máquina alvo e o seu o sistema operacional, a linguagem a ser utilizada deve explorar as características do ambiente definido: concorrência e multitarefa.

Analisamos algumas opções entre as linguagens concorrentes: Módula 2, ADA, Pascal concorrente, CHILL e consideramos também o uso de linguagens convencionais: C e Pascal. Estas últimas necessitavam do nosso compromisso de implementarmos a interface com as primitivas do sistema operacional SOP. Das linguagens concorrentes concluímos que a melhor opção era escolher a linguagem CHILL pois seus mecanismos de concorrência são suportados pelo sistema operacional SOP.

CHILL (CCITT High Level Language) é uma linguagem recomendada pelo CCITT, na programação de sistemas de comutação telefônica [CHILL 7].

As principais características da linguagem são:

- suporta aplicações de tempo real provendo mecanismos para comunicação e sincronização de processos e supervisão de tempo;
- permite programação concorrente em arquiteturas mono ou multiprocessadoras;
- possui mecanismos de compilação em partes;
- permite programação altamente modularizada e estruturada;

- permite o conceito de tipos abstratos de dados;
- oferece mecanismos de tratamento de exceções.

4 Implementação de protocolos usando CHILL

O sub-sistema de comunicação pode ser implementado como um conjunto de tarefas concorrentes que se comunicam, e são gerenciadas, por um núcleo em tempo real. Este núcleo é responsável também pelo tratamento de interrupções (temporizações, por exemplo).

A linguagem CHILL oferece vários recursos que facilitam a implementação de protocolos [CHILL 8]. A seguir serão descritos em detalhes cada um deles, bem como seus modos de utilização.

4.1 Processos

Quando se adota o modelo de camadas para especificar um protocolo de comunicação, assume-se que cada camada é uma entidade independente que oferece à camada superior um conjunto de serviços, e utiliza os serviços prestados por sua camada inferior.

Portanto, cada camada deve ser implementada como uma tarefa independente e o funcionamento do protocolo de comunicação baseia-se na execução de todas as tarefas (correspondentes às camadas) de modo concorrente.

A linguagem CHILL oferece o conceito de processo. Por definição, um processo é uma unidade de execução originada pela ativação de uma seqüência de código e dados. A sintaxe para a definição de um processo é:

```
<Nome do processo>:

PROCESS ( <Lista de parametros>);

  <Comandos>

END;
```

Este recurso possui importantes facilidades: mecanismos de sincronização e comunicação de dados entre processos.

4.2 Comunicação e sincronização entre processos

As camadas de um protocolo de comunicação se comunicam trocando serviços. Portanto fazem-se necessários mecanismos que permitam a comunicação de dados entre processos. A linguagem CHILL oferece quatro opções:

- Evento: Sincroniza execuções dos processos através de instruções DELAY (aguardar por um evento) e CONTINUE (liberar os processos que aguardam por um determinado evento).

- Região: Organiza o acesso a uma área comum de dados compartilhada por vários processos. Uma vez que um processo utilize recursos pertencentes a uma região, esta é reservada e se outros processos tentarem o acesso a ela eles ficarão bloqueados até que a região seja liberada.

- Sinal: Provê sincronização e comunicação independente da localização dos dados. Ele permite a composição e a decomposição de um conjunto de valores a ser transmitido. A instrução SEND é utilizada para enviar uma lista de valores (de quanquer tipo) e a instrução RECEIVE CASE é utilizada para identificar um sinal e seus valores associados.

- "Buffer": Provê um meio de sincronização e comunicação através de um conjunto de elementos do mesmo tipo. As operações são utilizadas através das instruções SEND e RECEIVE CASE.

No caso de protocolos de comunicação o uso de sinais é muito interessante, pois pode-se utilizar o nome do sinal para identificar o serviço, e os parâmetros, para especificar os parâmetros do serviço. O envio de sinal não causa a suspensão do processo e o sincronismo surge naturalmente: um processo é suspenso quando espera por sinais e sua fila está vazia; e automaticamente ativado quando recebe um sinal.

Além disso, o uso de sinais apresenta vantagens sobre os demais recursos porque: evento só permite sincronização; região necessita esforços adicionais para identificar o serviço e definição de estruturas de dados para os parâmetros; e finalmente os "buffers" tem a restrição de armazenar elementos do mesmo tipo.

A camada *N* que necessita solicitar um serviço da camada *N-1* envia um sinal a ela indicando o serviço desejado:

```
SEND  <Sinal> TO  <Processo>
```

A camada *N-1* que oferece este serviço identifica o sinal e providencia as atitudes apropriadas a cada sinal, da seguinte maneira:

```
RECEIVE CASE SET <Processo>;

( <Sinal> IN <lista de parametros> ): BEGIN
                                        <Comandos>
                                      END;
ELSE <Comandos>;

ESAC;
```

Além de permitir o sincronismo entre os processos, é possível associar a cada sinal um nível de prioridade que será levado em conta no tratamento deste. Isto é muito adequado para o caso onde algum serviço deve ser priorizado sobre outros. Exemplo: temporizações.

A linguagem também permite o envio de sinais a processos localizados em diferentes processadores. Para o projeto RALFO, que possui várias placas processadoras, é particularmente importante.

4.3 Temporizações

A linguagem CHILL oferece facilidades de temporização e agendamento nos mecanismos de comunicação. A finalidade básica é permitir a ativação de processos e envio de sinais nos vencimentos de tempos solicitados [CHILL 6].

Pode-se associar temporização às ações: SLEEP (onde após o vencimento do tempo associado, o processo é ativado); DELAY e RECEIVE (uma exceção TIME-OUT é causada pelo vencimento do tempo); e SEND (um sinal é enviado ao final da temporização).

A função de agenda permite ativar processos (SLEEP) ou enviar sinal (SEND) no vencimento de uma data ou horário previamente agendado. Pode-se estabelecer intervalos de horários onde o sinal será enviado em intervalos pré-definidos.

Existem funções que permitem cancelar ou reiniciar uma temporização ou agendamento.

A seguir, ilustraremos com um exemplo, a utilização de um sinal temporizado:

```
DCL
   id_A, id_B INSTANCE;

SIGNAL
   B TO Proc_B;


Proc_A:
PROCESS ();
 DCL
    tmr temporization,
    pt_timelocation ref_timelocation,
    var_timelocation time_location := [1,1,0];

  /* Configuracao do Temporizador */

  pt_timelocation:= -> var_timelocation;
  tmr:=[.tempkind: temp ,
        .timeloc: pt_timelocation,
        .pe: [.after: [.value:2, .unit:sec],
```

```
                .every: [.value:5, .unit:sec],
                .during:[.value:60,.unit:sec] ] ];

  Timed_Send (tmr);
  SEND B TO id_B;

END Proc_A;

Proc_B:
PROCESS ();
   DO FOR EVER;
    RECEIVE CASE
      (B): PUTF(TTY, 'RECEBIDO SINAL B');
    ESAC;
   OD;
END Proc_B;

id_A:=START Proc_A();
id_B:=START Proc_B();
```

Neste exemplo um sinal B será enviado pelo processo Proc_A ao processo Proc_B durante 60 segundos a cada 5 segundos, depois de 2 segundos da execução da instrução SEND.

No caso de protocolos de comunicação, onde é muito comum o uso de temporizadores, pode-se definir um processo temporizador que seria ativado quando recebesse uma instrução para ativar um determinado temporizador. A partir daí, enviaria ao processo solicitante, em intervalos definidos para cada temporizador, um sinal de expiração, até que este cancelasse o pedido de temporização.

4.4 Modularidade

A implementção de todas as camadas de um protocolo de comunicação pode exigir um esforço muito grande de programação e grande quantidade de linhas de código. Uma vez que cada camada funciona como uma tarefa independente, ela pode ser implementada de forma independente também. Para isso são necessários recursos que garantam modularidade e integridade no desenvolvimento de software.

Módulos constituem a unidade básica de estruturação de programas CHILL. Eles oferecem controle de visibilidade dos objetos; encapsulamento e abstração de dados; decomposição de programas em partes que podem ser desenvolvidas de maneira segura e independente; e acesso mutuamente exclusivo dos processos concorrentes aos objetos declarados localmente [APCC 9].

O módulo define o escopo da visibilidade dos objetos. Os nomes criados dentro de um módulo não são visíveis fora dele e vice-versa. Essa visibilidade pode ser alterada através de instruções de importação (SEIZE) ou exportação (GRANT) de objetos.

A independência na programação é conseguida através do mecanismo de programação em partes: a idéia é que quando uma parte do programa vai ser desenvolvida separadamente, ela seja substituída por um módulo de especificação , que define as propriedades estáticas dos nomes exportados por ele. Por exemplo:

Considere um programa esquematizado da seguinte maneira:

```
Prog1: MODULE
          .
          .
          .
          Prog2: MODULE
                    .
                    .
                    .
                   END Prog2;
       END Prog1;
```

Suponha que se deseja desenvolver Prog2 separadamente. O programa anterior deve ficar da seguinte forma:

```
Prog1: MODULE
          .
          .
          .
          Prog2: SPEC MODULE
                    .
                    .
                    .
                   END Prog2;
       END Prog1;
```

O corpo do módulo Prog2 só deverá conter os objetos visíveis ao módulo Prog1 (variáveis e definições de procedimentos, por exemplo) e neste contexto é denominado módulo de especificação. Assim o módulo Prog1 pode ser desenvolvido em paralelo com o Prog2. A implementação efetiva do módulo Prog2 é feita através de outra definição do módulo, porém sem a palavra SPEC, indicando que é a implementação real.

```
CONTEXT
   .
   .
   .
```

```
FOR
Prog2: MODULE
          .
          .
          .
       END Prog2;
```

O contexto define os nomes que serão importados por este módulo e aqueles que serão exportados por ele.

Pode-se compilar os dois módulos independentemente, pois a interface entre eles já está estabelecida através do módulo de especificação.

4.5 Implementação da camada LLC da RALFO

É discutida a seguir a implementação da camada LLC da RALFO utilizando a linguagem CHILL. A camada LLC tem como camadas vizinhas as camadas Aplicação (camada superior) e MAC (camada inferior). Ela oferece 5 tipos de serviço:

- Estabelecimento de conexão: L_Connect;
- Transferência de dados: L_Data_Connect;
- Desconexão: L_Disconect;
- Reiniciação do enlace: L_Reset;
- Controle de fluxo do enlace: L_Connection_FlowControl;

e utliza o serviço de envio de dados (MA_Data) oferecido pela camada MAC.

A seguir esquematizaremos um trecho do corpo principal do protocolo:

```
Protocolo_LLC:
MODULE

SIGNAL
/*  Servicos ofidos pela LLC a camada superior          */
  LCr <> SID = 01 <> =(Tipo_la, Tipo_ra, Tipo_sc),
  LDCr <> SID = 02 <> =(Tipo_la, Tipo_ra, Tipo_l_sdu),
  LDr <> SID = 03 <> =( Tipo_la, Tipo_ra),
  LRr <> SID = 04 <> =( Tipo_la, Tipo_ra),
  LCFCr <> SID = 05 <> =( Tipo_la, Tipo_ra, Tipo_a),

/*  Servicos oferecidos pela LLC a camada MAC       */
  MADi <> SID = 016 <> =( Tipo_da, Tipo_sa, Tipo_m_sdu, Tipo_rs, Tipo_rsc),
  MADc <> SID = 017 <> =( Tipo_ts, Tipo_psc),
```

```
LLC:
PROCESS ();

/* Iniciar a tabela de enlaces da camada LLC */
Inicializa_Tabela ();

/* Definicao dos enderecos dos processos que executam camadas vizinhas a LLC */
Proc_APL:= Enderecos(Maquina).APL;
Proc_MAC:= Enderecos(Maquina).MAC;

/* Tratamento dos sinais recebidos */

DO FOR EVER;

  RECEIVE CASE SET Proc_Origem;

/* Servicos solicitados pela camada superior */

  ( LCr   IN Lla, Lra, Lsc):
    BEGIN Evento := eLCr; la:= Lla; ra:= Lra; sc:=Lsc; END;
  ( LDCr  IN Lla, Lra, Ll_sdu):
    BEGIN Evento := eLDCr; la:= Lla; ra:= Lra; l_sdu:=Ll_sdu; END;
  ( LDr   IN Lla, Lra):
    BEGIN Evento := eLDr; la:= Lla; ra:= Lra; END;
  ( LRr   IN Lla, Lra):
    BEGIN Evento := eLRr; la:= Lla; ra:= Lra; END;
  ( LCFCr IN Lla, Lra, La ):
    BEGIN Evento := eLCFCr; la:= Lla; ra:= Lra; a:=La; END;

/*  Servicos solicitado pela camada MAC  */

  ( MADi IN Lda, Lsa, Lm_sdu, Lrs, Lrsc):
    BEGIN Evento := eMADi; da:= Lsa; sa:= Lda; m_sdu:=Lm_sdu;
          rs:=Lrs; rsc:=Lrsc; END;

  ELSE  PUTFLN(TTY,'ERRO-LLC: Primitiva nao esperada ');

ESAC;


/* Procurar o enlace na tabela de enlaces do processo LLC  Status indica o */
/* sucesso da operacao.                                                     */
  Procura_Enlace(SSAP,DSAP,Enlace,Status);
```

```
/* Caso o enlace nao se encontre na tabela ele pode ser um candidato a enlace */
  IF NOT(Status) THEN Enlace.Estado:= ADM;   FI;

/* Tratamento do enlace segundo o seu estado no protocolo LLC  */
   CASE Enlace.Estado OF
   (ADM): Estado_ADM (Enlace);
   (SETUP): Estado_SETUP (Enlace);
   (NORMAL): Estado_NORMAL (Enlace);
   (ERROR): Estado_ERROR (Enlace);
   (RESET): Estado_RESET (Enlace);
   (D_CONN): Estado_DCONN (Enlace);
   ESAC;

/* Atualizacao do enlace na tabela de enlaces */
   Atualiza_Enlace(SSAP,DSAP,Enlace,Status);
   IF NOT Status THEN Erro (0); FI;

END LLC;

END protocolo_LLC;
```

Segue abaixo um trecho do tratamento do estado ADM do protocolo LLC conforme seu diagrama de transições:

```
Estado_ADM:
PROC (Enlace Tipo_Enlace LOC);

DCL
   Status_Cria_Enlace BOOL,
   P Tipo_Flag,
   PDU  Tipo_m_sdu;

 CASE Evento OF
   (eMADi): BEGIN
             CASE Id_PDU OF
              (SABME): BEGIN
                       IF (Classe_PDU=CMD) AND (PF=B0) THEN
                        IF Aceita_Conexao() THEN /* E' possivel aceitar a
                                                    conexao */
                        /* Criar uma entrada na tabela de enlaces */
                        /* Atualiza a variavel local SSAP */
                        Cria_Enlace(SSAP,DSAP,Enlace,Status_Cria_Enlace);
                        IF Status_Cria_Enlace THEN
                          la.LSAP:=SSAP;          Enlace.VS :=0;
```

```
                        Enlace.VR :=0;          Enlace.P_Flag :=B0;
                        Enlace.Remote_busy:= B0; Enviar_LCi();
                        Enviar_UA(RSP,PF);      Enlace.Estado:=NORMAL;
                        Enlace.Subestado:=NONE;
                       ELSE Enviar_DM(RSP,PF);
                       FI;
                      ELSE  /* Nao e' possivel aceitar a conexao */
                       Enviar_DM(RSP,PF);
                      FI;
                     FI;
                     END;
            (DISC): BEGIN;
                     IF Classe_PDU=CMD THEN Enviar_DM(RSP,PF);
                     ELSE Erro(1); FI;
                     END;
            ELSE     BEGIN;
                     IF (Classe_PDU=CMD) AND (PF=B1)
                     THEN Enviar_DM(RSP,PF);
                     ELSE Erro(1);
                     FI;
                     END;
          ESAC;
         END;
  ELSE ;
 ESAC;
END Estado_ADM;
```

O envio de um sinal (SABME) foi implementado da seguinte forma:

```
Enviar_SABME: /* Quadro tipo U */
PROC (CR Tipo_Classe_PDU, PF Tipo_Flag);
  DCL PDU Tipo_m_sdu;

  PDU.DSAP:= Converte_INT_BIN(DSAP); PDU.SSAP:= Converte_INT_BIN(SSAP);
  IF CR=RSP THEN PDU.SSAP := PDU.SSAP OR B'1000_0000'; FI;
  IF PF=B1 THEN PDU.Controle:=B'1111_1110_0000_0000';
           ELSE PDU.Controle:=B'1111_0110_0000_0000';
  FI;
  PDU.Info:= (16)B'0';
  SEND MADr (da, PDU, Prioridade) TO  Proc_MAC;
END Enviar_SABME;
```

Observe que uma vez tendo o protocolo especificado através de diagramas de transições para cada estado do protocolo, a implementação fica imediata:

1. Define-se um processo para cada camada.
2. Para cada camada define-se um sinal para cada primitiva.
3. Para cada estado de uma camada define-se um procedimento que fará o tratamento dos sinais recebidos através de instruções do tipo CASE.
4. Todo o envio de sinais é feito através de instruções SEND.

Todos os aspectos da implementação da comunicação e sincronização dos processos ficam transparentes ao programador, tendo este que se preocupar exclusivamente com o protocolo de comunicação.

4.6 Testes de funcionamento

Para testar o funcionamento da camada LLC definimos um ambiente de teste no próprio PP. Para isto implementamos parcialmente as camadas Aplicação e MAC, preocupando-nos principalmente com suas interfaces de comunicação com a camada LLC.

Para simular dois nós em comunicação ativamos dois processos para cada camada: um fazendo o papel de uma camada no nó local, e o outro, no nó remoto. Como o hardware e o software que implementam o controle de acesso ao meio físico não estavam prontos, estabelecemos que a camada MAC local se comunicaria diretamente com a MAC remota. Assim, através das seis camadas em execução: Aplicação local e remota, LLC local e remota, MAC local e remota; foi possível testar todo o funcionamento da camada LLC sem ter que esperar pelo desenvolvimento das demais camadas.

5 Conclusões

Neste trabalho foi discutida a implementação do protocolo de comunicação na rede com integração de serviço denominada RALFO, utilizando a linguagem CHILL.

O uso da linguagem CHILL simplificou muito o desenvolvimento do protocolo, pois ela contém recursos para o desenvolvimento de aplicações em tempo real que necessitam de primitivas de supervisão de tempo e concorrência. Constatou-se que é uma linguagem poderosa, rica em recursos e fácil de ser utilizada.

Enfim, a linguagem CHILL se mostrou muito adequada para o desenvolvimento de protocolos de comunicação em uma rede de serviços integrados, principalmente no caso da RALFO onde o ambiente é multitarefa e multiprocessador. Como atividade futura, pretendemos verificar se a implementação está adequada em relação ao desempenho de uma rede integrada.

Referências

[PP 1] "Processador Preferencial - Relatório Interno, CPqD - Telebrás, 1984.

[SOP 2] " Especificação do Sistema Operacional PP-SO/P - Relatório Interno, CPqD - Telebrás, 1990.

[Pessoa 3] Pessoa P.M.C.: "Implementação de uma interface de voz para rede local com fibras ópticas e integração de voz e dados", dezembro 1991. Tese de mestrado apresentada na FEE, UNICAMP.

[Guardi 4] Guardiero P.R.: "Um método de controle de acesso para rede local com fibras ópticas e integração de voz e dados", dezembro 1991. Tese de doutorado apresentada na FEE, UNICAMP.

[Carnei 5] Carneiro M.C.C.: "Rede local de computadores multi-serviços: proposição de uma arquitetura e implementação da camada sinalização", 1991. Tese de mestrado apresentada na Universidade Federal de São Carlos, SP.

[CHILL 6] "Manual de Programação CHILL para o PP-SO/P" CPqD - Telebrás, 1990.

[CHILL 7] "CCITT High Level Language (CHILL) - Recommendation Z.200 - ISO/IEC 9496", CCITT, 1988.

[CHILL 8] Sobrinho S.C., Araújo C.C., Luz P.R., Cavalli E., Rolim L.A.: "A Real-Time Operating System for Distributed Applications with CHILL Execution Environment Support", "Proceedings of the 5th CHILL Conference - Rio de Janeiro, Brazil, 19-22 March, North Holland, 1991".

[APCC 9] "APCC 4.0 - Documentação do Usuário", Volumes I e II, CPqD Telebrás, 1991.