Utilização de uma Metodologia Baseada em Objetos na Implementação de uma Rede Local de Custo Zero

Márcio Quintaes Marchini - UFSC
Gláucio Estanislao Schumacher - UFSC

GRUPPOO - Grupo de Pesquisa em Programação Orientada a Objetos
EDUGRAF - Laboratório de Software Educacional e Gráfico
CEC/CTC/UFSC - Universidade Federal De Santa Catarina

RESUMO

O objetivo deste trabalho é apresentar o desenvolvimento de uma rede local de custo zero. A metodologia de desenvolvimento é baseada em objetos utilizando a linguagem Modula-2. É utilizado um mecanismo de metáforas na concepção das classes de objetos que constituem a arquitetura do sistema.

Inicialmente é demonstrado o mecanismo de metáforas . Em seguida, cada uma das classes abstraídas é detalhada. Finalmente, apresenta-se toda a arquitetura do sistema e o seu funcionamento.

ABSTRACT

The aim of this paper is to show the development of a low-cost local network. The methodology used is Object Based, using the Modula-2 language. A mechanism of metaphors is used to conceive the classes of objects that compose the system framework.

First we describe the mechanism of metaphors. We also describe each of the classes conceived. At last, we present the system architecture and how it works.

1 - Introdução

Compartilhamento e integração de recursos e serviços são os grandes atrativos de redes de computadores. Redes locais são especialmente interessantes em pequenas organizações e laboratórios, principalmente em pesquisa.

Com a evolução dos estudos feitos pelo nosso grupo na área de hipertexto/hipermídia, sentiu-se a necessidade de uma rede local para a implantação de um sistema de hipertexto distribuído.

A falta de recursos evidenciou que a solução seria a implantação de uma rede local de custo zero. Para que isso fosse possível, teríamos que nos utilizar do recurso de hardware que possibilitasse a comunicação e acompanhasse os equipamentos disponíveis: a interface serial RS-232 [Itautec88].

Software passou a ser o ponto central, e para viabilizá-

lo teríamos que escolher uma metodologia (por nós considerada) altamente eficiente.

A construção foi baseada no paradigma de objetos [Meyer88] e este trabalho relata a forma de abstração das classes de objetos e a forma de implementação em uma linguagem de programação.

2 - Metáforas na Concepção de Classes

É só olhar em nossa volta e podemos perceber um mundo rico em objetos. Carros, casas, canetas, mesas, etc , todos fazendo parte do chamado "mundo real".

Ao analisarmos o "mundo computacional", pronto... lá se foram embora os nossos objetos ! A diferença de riqueza dos dois mundos parece ser tão grande que poderia até ser desanimadora.

Ao abstrairmos um modelo mental a partir do modelo do "mundo real", há um "gap" semântico muito grande para a sua respectiva implementação - o chamado modelo computacional.

Uma metodologia que auxilia na concepção das classes de objetos que consideramos interessante é o uso de metáforas [Laurel90]. Mas o que vem a ser isto ?

Quando construímos modelos computacionais, muitas vezes eles se assemelham em muito com modelos reais, embora muitas diferenças possam existir. Outras vezes, construímos modelos tão distantes dos modelos reais aos quais estamos acostumados que fica difícil abstrair, de forma intuitiva, a sua funcionalidade.

Para diminuir este "gap", usamos a metáfora. Na verdade, o que fazemos é buscar um modelo real que mais se assemelhe ao modelo computacional que queremos construir. O modelo real serve de paralelo, orientando a funcionalidade do nosso modelo computacional. Isso não significa que nosso modelo deva ser rigorosamente igual ao modelo real. Afinal, o uso do computador nos permite graus de liberdade muito grande, permitindo-nos estabelecer modelos que superem as limitações dos seus correspondentes no mundo real.

Buscar objetos do mundo real para mapeá-los para o mundo computacional ajuda a diminuir o "gap" para aquele que implementa bem como para aquele que reutiliza implementações feitas por terceiros.

3 - Uma Metodologia de Implementação

Além de uma forma de capturar conhecimento suficiente do modelo real, é preciso que tenhamos uma boa metodologia de mapeamento para um modelo computacional.

No nosso caso, utilizamos uma metodologia chamada GRUPPOOTECA [Melgarejo88], desenvolvida também pelo nosso

grupo. A sua descrição mereceria , por si só, um outro artigo. Assim, daremos apenas aspectos bem gerais.

Basicamente, ela consiste de:

- Uma linguagem de programação : Modula-2 [Wirth85] [Messer86]
- Uma metodologia de implementação de classes de objetos, sem o suporte de herança [Meyer88]
- Uma metodologia de genericidade em Modula-2 [Melgarejo88]
- Uma filosofia de programação [Melgarejo88]

4 - A Implementação da Rede

A GRUPPOOTECA nos pareceu bastante adequada para a implementação da rede devido, principalmente, a:

1) Modula-2 ter sido projetada com recursos de baixo nível que viabilizam a implementação até mesmo de sistemas operacionais. Dentre estes recursos está o mecanismo para tratamento de interrupções, o que seria vital no nosso projeto

2) O fato de a GRUPPOOTECA oferecer mecanismos de abstração e genericidade que tornariam todo o nosso código reusável.

3) A possibilidade de incorporação da rede à própria biblioteca de classes da GRUPPOOTECA, permitindo que novas aplicações pudessem ser desenvolvidas de forma a funcionarem em rede.

5 - Identificação das Classes de Objetos

5.1 - Time-Out

Uma primeira preocupação do nosso projeto foi com o mecanismo de Time-out, o qual se faz necessário em qualquer implementação de uma rede. Como melhor definir um mecanismo de Time-out ?

A primeira preocupação é abstrair a funcionalidade de um mecanismo de time-out. Tal mecanismo é usado quando desejamos, depois do esgotamento de um intervalo de tempo, realizar alguma coisa. Por exemplo, poderíamos ter um mecanismo de time-out iniciado para ecoar um beep no computador caso o usuário demore mais do que trinta segundos para digitar uma tecla.

Bastou olhar à nossa volta para achar um objeto do mundo real que tinha funcionalidade muito parecida com a do modelo que tínhamos em mente : o cronômetro regressivo com alarme. Tais cronômetros permitem que você programe um intervalo de tempo que fará com que o cronômetro soe o alarme após esgotado tal intervalo. Queríamos, então, um cronômetro regressivo com alarme, o qual poderíamos instanciar com o intervalo de tempo

que precisássemos (ver figura 1).

Figura 1 : Objeto Cronômetro Regressivo com Alarme

O problema passou a ser a forma de captar a metáfora do alarme. Num cronômetro real, temos uma música ou uma campainha como alarme. No nosso caso queríamos que o "alarme" também pudesse ser estabelecido pelo usuário da classe. Queríamos uma forma de criar cronômetros genéricos, tanto quanto ao tempo quanto à semântica do alarme. Queríamos um alarme que pudesse variar desde um beep até a retransmissão de um pacote pela rede.

Lá fomos nós atrás de outro modelo do mundo real...

Queríamos que o alarme fosse capaz de capturar a idéia de dinamicidade. Queríamos que ele refletisse uma ação a ser executada. No nosso modelo de objetos, queríamos na verdade uma mensagem e um objeto. Quando o alarme tocasse, a mensagem seria enviada ao objeto, desencadeando um acontecimento. Um exemplo seria o console como objeto e beep como mensagem. Quando o alarme tocasse, seria enviada a mensagem beep ao objeto console, fazendo soar o sinal.

Faltava captar um modelo que representasse esta idéia, bem como a possibilidade de configuração (mudar o objeto, mudar a mensagem, ou mudar ambos).

5.2 - Chips

Abstraímos a idéia de chip. No nosso modelo computacional, um chip seria composto por um objeto e uma mensagem a ser enviada a este objeto (ver figura 2).

Figura 2 : Objeto do Chip

O chip poderia ser ativado, o que faria com que a mensagem fosse enviada ao objeto e alguma ação se desencadeasse. Assim, um chip "sonoro" poderia ser instanciado a partir do objeto console e da mensagem beep. Quando o chip fosse ativado, a mensagem beep seria enviada ao console.

Assim, nosso cronômetro regressivo com alarme seria programável e configurável ao decorrer de sua existência. A um

determinado momento, o chip do cronômetro poderia ser mudado, para que ele, por exemplo, transmitisse uma mensagem pela rede. Assim, o mesmo cronômetro poderia ser reconfigurado para ter uma funcionalidade bem diferente da original. Se este não fosse o interesse, uma forma alternativa seria a instanciação de um outro cronômetro regressivo com alarme, contendo um outro chip.

Agora , a GRUPPOOTECA tinha enriquecido com duas novas classes : chips e cronômetros regressivos com alarme.

Aqui podemos notar o significado do termo metáfora. Não queríamos mapear a idéia de alarme exatamente como no mundo real. Alarme, no nosso modelo, seria a ativação de um chip, o qual poderia ser trocado e reconfigurado a qualquer momento.

6 - O Meio Físico

Conforme dito anteriormente, o meio físico a ser utilizado como transmissão de dados seria a interface serial e fios do tipo par trançado. Já tínhamos o modelo real (placas RS 232 existem !) , e queríamos enriquecê-lo no nosso modelo computacional.

Uma característica do nosso modelo seria a possibilidade de instanciar placas (interfaces) seriais no próprio programa. Assim, placas seriais seriam objetos como outros quaisquer na GRUPPOOTECA (ex. filas, conjuntos, chips, etc). O programador se comunicaria com tais placas através de mensagens, como no mundo de objetos.

A placa oferece serviços de interrupção, mas queríamos elevar o nível desta abstração de serviço. Buscávamos uma metáfora que nos parecesse mais natural e alto nível. A idéia foi a mesma encontrada no caso dos nossos cronômetros regressivos com alarme : utilização de chips.

Nossa placa seria programável quanto à velocidade de transmissão, número de stop bits, quantos bits de paridade, etc. Mas seria configurável (reprogramável) através de chips (ver figura 3).

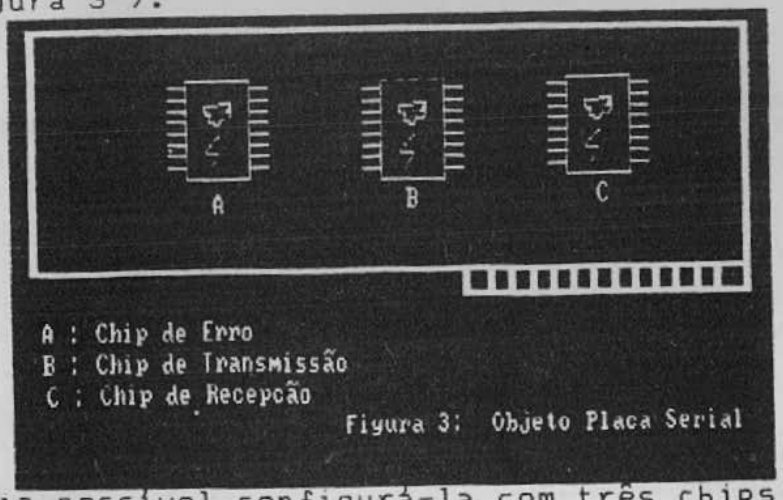

Figura 3: Objeto Placa Serial

Seria possível configurá-la com três chips diferentes :

- O primeiro seria ativado quando um caracter chegasse na interface serial através do meio físico
- O segundo seria ativado quando um erro ocorresse (erro de paridade, "overrun" , "break", etc)
- O terceiro seria ativado quando a interface estivesse pronta para transmitir um caracter

Relembrando do nosso chip "sonoro", poderíamos fornecê-lo à nossa placa serial, de forma que ele soasse um beep no console sempre que um caracter chegasse na placa serial.

Um segundo chip "sonoro" poderia ser fornecido, talvez com outro tipo de som , para que uma buzina tocasse quando houvesse um erro dentre aqueles mencionados anteriormente.

Um terceiro chip poderia fazer algo qualquer quando a interface estivesse disponível para uma próxima transmissão.

Agora, a GRUPPOOTECA tinha mais uma classe de objetos - placa serial - a qual já reusava uma outra classe desenvolvida anteriormente : chips.

Optamos por um meio físico em barramento [Tanenbaum88], e para que isso fosse possível, bastava colocar em curto os pinos de transmissão e recepção de dados do conector serial (Tx e Rx respectivamente).

7 - Um Método de Acesso ao Meio

O próximo passo foi escolher um método de acesso ao meio [Tanenbaum88] [Giozza86] [Tarouco86]. Nossa escolha foi o método CSMA/CD PR [Giozza86] , o qual dá prioridade para as mensagens do tipo "Acknowledgement" (ACK) , evitando suas colisões com mensagens normais. Assim, um melhor aproveitamento do meio físico poderia ser feito.

8 - Metáforas da Rede

Faltava definir uma metáfora para o tipo de informação básica que iria fluir no meio, bem como outra pra definir quem iria prover este tipo de serviço.

Queríamos abstrair a idéia de quadros ou pacotes que seriam despachados pela rede, bem como a entidade (classe de objeto, na verdade) que iria nos fornecer este tipo de serviço.

Mais uma vez, observando o mundo à nossa volta, pudemos abstrair uma idéia similar àquela buscada por nós : o serviço de correio.

No correio, mandamos objetos (cartas, pacotes, etc) que deverão chegar a um destinatário por nós especificado.

Em uma carta, por exemplo, temos o remetente, o destinatário, e o conteúdo da carta propriamente dito. Em um pacote (de correio ainda), temos as mesmas informações.

Mas num serviço de rede, muitas vezes precisamos ter certeza que o destinatário recebeu a nossa informação despachada.

No serviço de correios encontramos uma idéia bastante similar, que é o serviço de encomendas com resposta garantida (carta registrada, por exemplo).

Voltando ao nosso modelo computacional, queríamos ter a possibilidade de mandar mensagens de difusão na rede, ou mensagens que se difundissem para um determinado grupo de pessoas na rede. O nosso modelo, no mundo real, não tinha esta idéia (fica difícil imaginar a idéia de enviar uma carta para todos os habitantes de São Paulo, por exemplo).

Assim, nosso modelo seria mais uma vez baseado no modelo real (correio) , mas implantado de forma a adicionar-lhe funcionalidade, tornando-o mais rico. Desta forma, seguiríamos a nossa metodologia de metáforas.

9 - Cartas

Demos o nome de carta à esta informação que iria circular pela rede. A carta, além de seu conteúdo, teria a especificação de um remetente e um destinatário. Mapeamos endereços de remetentes e destinatários como sendo números (ver figura 4). Números em uma determinada faixa seriam endereços individuais, e em outra faixa seriam endereços de grupo. Assim, cartas cujo destinatário fosse um grupo, não teriam o serviço de confirmação automático do nosso correio. Cartas endereçadas a uma pessoa (estação) específica, teriam o serviço de confirmação.

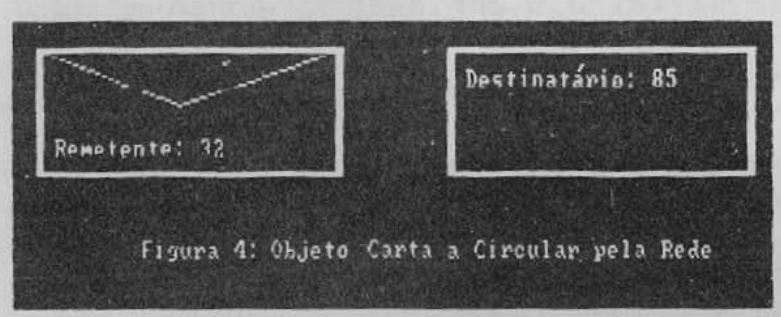

Figura 4: Objeto Carta a Circular pela Rede

Em outras palavras, quando uma carta fosse enviada, o remetente esperaria uma confirmação do destinatário somente se este não fosse um grupo de estações. No caso de ser um endereço de grupo, o remetente teria a confirmação de que sua correspondência foi enviada, sem ter, porém, a confirmação de que foi recebida.

O mecanismo de difusão (para todas as estações) seria assim implementado, fazendo com que todas as estações fizessem parte de um grupo, o chamado grupo de difusão. Assim, uma mensagem destinada a este grupo seria recebida por todas as estações da rede (inclusive a que enviou a mensagem !).

O dimensionamento da carta foi feito de forma a impedir que uma estação monopolizasse o meio. Assim, não seria permitido a uma estação utilizar o meio de forma muito prolongada. Desta forma, as cartas teriam que ter um tamanho máximo, o qual deveria causar a utilização do meio por um tempo razoavelmente curto. Este tempo foi por nós estipulado (através de diversos testes e comparações) em cerca de um segundo. Nossa

carta, a este nível, seria limitada a um tamanho máximo permitido.

O que aconteceria se desejássemos transmitir um arquivo, por exemplo, cujo tamanho fosse muito superior ao tamanho máximo suportado pela rede ?

Devido a nossa metodologia, abstraímos um nível mais alto de serviço, o qual seria fornecido ao usuário final da rede. Ele se utilizaria de um agente que se encarregaria de despachar qualquer tipo de informação para ele, qualquer fosse o seu tamanho. Esta abstração foi chamada de Posto de Correio. Ele seria o responsável pela fragmentação da informação em unidades menores (cartas), as quais seriam despachadas pela rede, utilizando os carteiros. Assim, o problema da limitação do tamanho máximo seria resolvido com mais um nível, o qual seria encarregado desta fragmentação. Assim para o usuário final, a rede estaria transmitindo informações de qualquer tamanho. Uma vez que tais pacotes de informação seriam fragmentados, dois usuários transmitindo um enorme conjunto de informação teriam a ilusão de paralelismo na rede, embora tal "paralelismo" fosse uma ilusão causada pela intercalação dos fragmentos (cartas) no meio físico.

10 - Carteiros

Batizamos de carteiro a entidade responsável por despachar e nos entregar as cartas (ver figura 5). Ao instanciarmos um carteiro, precisávamos apenas lhe dizer qual o nosso endereço. Assim, cartas a nós endereçadas nos seriam entregues pelo nosso carteiro.

Figura 5: Objeto Carteiro da Rede

Faltava abstrair a idéia da entrega da carta. Uma primeira solução seria a criação de uma caixa de correspondência, onde o carteiro automaticamente colocaria a nossa carta quando esta chegasse. Um problema nesta idéia é a semelhança que ela teria com os problemas do modelo real. Quando uma carta chega em minha casa, ela é colocada em minha caixa de correspondência. Porém, se me esqueço de olhar a caixa periodicamente, posso vir a ter que pagar contas com multas devido ao meu desleixo.

Assim, não queríamos obrigar ao usuário desta classe a necessidade de , periodicamente, verificar sua caixa de correspondência. Queríamos uma forma de avisá-lo quando a carta chegasse, para que ele fizesse dela o que bem entendesse.

Uma opção era o uso de processos em Modula-2. Assim, po-

deríamos ter uma comunicação entre os mesmos. Porém, nem todas as implementações suportam um mecanismo de escalonamento de processos. Além do mais, queríamos poder criar placas virtuais de comunicação (a ser explicado mais adiante), o que seria incompatível com o mecanismo de processos na implementação Modula-2 a nós disponível [TopSpeed88].

A metáfora escolhida foi a mesma empregada nos cronômetros regressivos com alarme e nas placas seriais : chips .

Assim, nosso carteiro seria programável e reconfigurável através de chips.

Seria possível configurá-lo com três chips diferentes :

- O primeiro seria ativado quando uma carta chegasse
- O segundo seria ativado quando um erro ocorresse (uma carta que exigia confirmação foi enviada e a confirmação não foi recebida ou então se o carteiro tentou mas não conseguiu enviar a carta)
- O terceiro seria ativado quando o carteiro estivesse livre (disponível) para enviar uma carta. Assim, poderíamos pedi-lo para nos chamar assim que ele enviasse a nossa carta. Desta forma poderíamos entregar-lhe uma outra carta para despachar.

É importante notar que a idéia de chips nos permitiria receber e transmitir cartas em "background". Desta forma, uma aplicação pode estar recebendo uma carta sem perceber. Ela será avisada somente quando a operação se completar (o mecanismo de interrupção é mascarado com a metáfora de "chips a serem ativados"). A transmissão também é feita em "background", fazendo com que a aplicação solicite o envio de uma carta e vá fazer uma outra tarefa. Aqui, novamente, o mecanismo baixo nível de interrupção é mascarado pela metáfora de chips. Desta forma, um mecanismo de processos é implementado via interrupções e mascarado via a nossa metodologia de metáforas.

Através de chips, a implementação do Posto de Correio seria similar. Ele fragmentaria um grande conjunto de informações em uma seqüência de cartas e solicitaria que o carteiro enviasse a primeira. Quando o carteiro terminasse, seu chip que indica disponibilidade seria ativado e o Posto de Correio poderia solicitá-lo a enviar a próxima carta da seqüência , e assim sucessivamente.

Na recepção, ele iria agrupando as cartas até formar a unidade de informação original. Quando esta estivesse pronta, ele poderia novamente ativar um chip, avisando ao usuário que um pacote chegou.

Assim, o Posto de Correio teria implementação bem similar ao nosso carteiro. Analisando os dois, é fácil notar esta similaridade:

- Um carteiro é responsável por fragmentar uma carta e enviá-la byte-a-byte , uma vez que esta é a unidade máxima de informação a ser transmitida pela placa serial. O carteiro também é responsável por entregar as cartas que chegarem e avisar quando ele não conseguir enviar uma carta.

- O Posto de Correio, que se utilizaria do nosso carteiro, seria responsável por fragmentar um pacote de informações (seja qual for o seu tamanho) e enviá-lo, carta-a-carta, uma vez que esta é a unidade máxima de informação a ser transmitida pelo carteiro. O Posto de Correio também é responsável por entregar os pacotes (mesmo que eles sejam constituídos de uma carta somente) que chegarem e avisar quando ele não conseguir enviar um pacote.

Assim, uma aplicação poderia optar por utilizar os serviços de um simples carteiro, respeitando suas limitações, ou de um serviço completo de correio (abstraindo-se e desconhecendo o fato de que o Posto de Correio se utiliza de carteiros) para enviar grandes quantidades de informações.

11 - Recepção e Transmissão das Cartas : CSMA/CD PR

Para que nosso carteiro pudesse transmitir as cartas, ele teria que se utilizar da interface serial (agora uma classe da GRUPPOOTECA). O problema passou a ser a especificação de como se dá esta comunicação: transmissor-receptor. Qual seria o protocolo de comunicação a ser feito a nível de bytes (byte-a-byte).

Esta comunicação passa por determinados estados (transmissão do endereço do remetente, transmissão do endereço do destinatário, etc). O segredo consiste em o carteiro ir evoluindo por estes estados, até completar o ciclo e retornar ao estado inicial. Cada estado tem uma semântica própria. Caso condições não sejam satisfeitas para um determinado estado, desconsidera-se o que foi recebido até então, fazendo com que o carteiro retorne ao estado inicial (pronto para receber). Um exemplo seria a chegada de um caracter errado à placa serial, o que faria com que esta ativasse seu chip de erro. Tal chip seria configurado com o objeto carteiro e a mensagem para retorná-lo ao estado inicial. Desta forma este erro evitaria que o carteiro evoluísse para o próximo estado, e sim retornasse ao estado inicial.

Há também um tempo máximo associado a cada estado. Um carteiro não pode permanecer em um determinado estado um período de tempo maior do que o tempo máximo especificado -no nosso caso, estipulamos 1 segundo.

Para implementar o mecanismo de temporização que faria com que o carteiro retornasse ao estado inicial caso um caracter demorasse muito tempo a chegar à placa serial, utilizamos o cronômetro regressivo com alarme. Assim, na realidade, o chip de chegada de caracter na serial iria:

1- Resetar o cronômetro regressivo com alarme (colocá-lo no seu valor inicial de programação).

2- Processar o caracter chegado.

No nosso caso, seria uma corrida contra o tempo. Enquanto o cronômetro regressivo vai diminuindo o seu valor, é processado o caracter e espera-se a chegada do novo caracter. Se o novo caracter chegar antes do alarme do cronômetro regressivo

tocar, este será reinicializado para o seu valor de programação e uma nova corrida contra o tempo se efetua. Caso o caracter demore para chegar, o cronômetro regressivo chegará a zero e ativará o chip, fazendo com que o carteiro retorne ao estado inicial.

Assim, o mecanismo de time-out é implementado através de um cronômetro regressivo com alarme. O chip deste cronômetro será instanciado como tendo o carteiro como objeto e uma mensagem especial a lhe ser enviada. Tal mensagem terá o efeito de retornar o carteiro ao estado inicial e reprogramar o cronômetro regressivo.

Assim, no caso de uma transmissão normal, o cronômetro nunca consegue soar o alarme e fazer com que o carteiro volte ao estado inicial. Isso porque o tempo do cronômetro é estipulado como sendo bem maior que o tempo necessário para fazer o processamento da chegada de cada caracter. Como a cada chegada de caracter o cronômetro é reinicializado, ele nunca chega a "tocar" em transmissões normais.

Caso a estação que transmite falhe durante o processo, todas as estações (as quais estão todas recebendo, uma vez que o meio é em barramento) irão retornar ao estado inicial devido aos respectivos cronômetros regressivos com alarme.

Um carteiro evolui de estados e , se ele reconhece a carta como sendo endereçada ao seu cliente (aplicação que o instanciou), ele instancia uma carta e, após montá-la, ativa o seu próprio chip de chegada de carta.

A aplicação pode, neste instante, pedir a carta ao carteiro e colocá-la numa caixa postal ou acordar um processo, por exemplo.

Caso um erro de transmissão ocorra no meio do processo (uma colisão, por exemplo) isso fará com que o chip da serial que é ativado quando ocorre erro entre em ação.

Este chip também é instanciado conforme o chip de time-out. Ele faz com que a estação receptora retorne ao estado inicial caso um erro (paridade, etc) ocorra durante a transmissão/recepção.

Uma das primitivas necessárias na técnica CSMA/CD PR é testar se o meio está livre. Uma vez que todas as estações recebem tudo aquilo que é transmitido, todas evoluem de estado. A diferença é que algumas apenas ignorarão a carta (por não lhes ter sido endereçada) e outras não. Assim, para saber se o meio está livre, basta testar se o estado do carteiro corresponde ao estado inicial. Assim, estar no estado inicial de recepção é uma pré-condição para transmissão. Estar neste estado significa que o meio está livre.

Agora que analisamos como se dá a recepção, precisamos analisar como se dá a transmissão.

Para que nossa técnica seja CSMA/CD PR, é preciso que detectemos colisões no instante que elas ocorrem. Além do mais, precisamos garantir que mensagens de reconhecimento (ACK) poderão ser transmitidas sem problemas (o meio estará garanti-

damente livre quando de sua transmissão).

Na transmissão, é inicializado um contador de tentativas (quantas vezes o carteiro tentará transmitir novamente caso ocorra colisões). Se esgotar o número de tentativas, o seu chip de erro será ativado.

O carteiro liga um cronômetro que o acordará de tempos em tempos. Na primeira vez, ele testa se o meio está livre. Caso não esteja, ele incrementa o contador de tentativas e reprograma o cronômetro para acordá-lo novamente após um determinado intervalo de tempo, o qual o carteiro julga suficiente para encontrar o meio livre (dimensionado como metade do tempo de transmissão de uma carta). Assim, de tempos em tempos ele testa a disponibilidade do meio, até que este esteja livre ou o número de tentativas se esgote.

Ao encontrar o meio livre o carteiro reprograma o cronômetro para acordá-lo após um tempo, chamado Tempo Básico de Espera na bibliografia sobre CSMA/CD PR [Giozza86]. É este o tempo que garante que uma mensagem de confirmação possa ser transmitida sem problemas.

Se após o tempo básico de espera o meio continua livre, o carteiro começará a transmitir. Na técnica CSMA/CD PR , é necessário que colisões sejam detectadas no instante que ocorrem. Assim, o alarme do chip de erro da serial fará com que o carteiro retorne ao estado inicial de transmissão caso ocorra colisão durante a transmissão da carta.

O estado do carteiro (transmissão) é atualizado e o caracter é transmitido. Após este caracter ser transmitido, o chip da serial que indica que ela está pronta para transmitir outro caracter será ativado e o processo pode ir se dando em cadeia. Isso porque este chip enviará uma mensagem ao carteiro que é responsável por atualizar o próprio estado e transmitir o próximo caracter. O processo se repete em cascata até chegar ao último caracter. O chip será novamente ativado, mas desta vez o carteiro não solicitará nada à serial. Ele ativará um outro cronômetro que o acordará depois de um intervalo de tempo. Este intervalo de tempo é aquele necessário para que uma mensagem de confirmação seja enviada (ACK). Ao ser acordado, o carteiro verifica se recebeu o ACK esperado. Caso positivo, ele retorna ao estado inicial de transmissão, estando apto para uma outra transmissão. Caso contrário, ele retorna ao estado inicial de transmissão mas ativa o seu chip de erro, para que a aplicação que dele se utiliza seja notificada.

12 - Utilizações

Uma qualidade que um item de software deve possuir é a reusabilidade [Meyer88]. Assim sendo, queríamos que a nossa rede fosse a mais reusável possível. Identificamos alguns tipos de utilização:

1) A utilização por um usuário da GRUPPOOTECA. Esta utilização seria trivial, uma vez que a aplicação teria simplesmente que ser cliente da classe que oferece serviços de rede.

Assim, por exemplo, uma aplicação pode ser desenvolvida em Modula-2, utilizando a GRUPPOOTECA, de forma que diversas destas aplicações possam se comunicar entre si. Um exemplo seria a implementação de um correio eletrônico. Outro exemplo, atualmente sendo desenvolvido por um dos usuários, é a implementação de um jogo de batalha naval em rede.

2) Uma outra forma de utilização seria para aqueles que, embora utilizando outro compilador, quisessem utilizar os serviços da rede por nós implementada. Assim, um programador utilizando uma outra linguagem poderia se utilizar da nossa rede. Isso pode ser feito através da implementação de um "driver" de rede, sob a forma de um programa residente.

3) Uma outra forma de utilização seria estender o sistema operacional da máquina utilizada de forma a ter comandos de rede. Assim, os usuários poderiam mover arquivos de uma máquina para a outra através da rede. Poderiam também ser oferecidos servidores de impressão e servidores de arquivo.

Uma solução encontrada para abranger o segundo e terceiro tipo é específica para equipamentos compatíveis com a linha IBM-PC.

A solução encontrada seria a criação de outra metáfora: a placa virtual de comunicação. Tal placa virtual seria instalada nos equipamentos que desejassem participar da rede, e aí então o suporte de desenvolvimento seria dado.

A placa virtual seria mapeada, para computadores IBM-PC, como sendo um programa residente. Tal programa ofereceria, através do conhecido serviço de interrupções do PC, a possibilidade que outras aplicações chamassem tais interrupções para prestar-lhes o serviço.

Para cobrir a necessidade do terceiro tipo, bastaria oferecer a placa virtual e alguns utilitários. Assim como o "FORMAT" estende o DOS para que ele possa formatar discos, poderíamos fornecer programas ("copie" e "diretório" por exemplo) os quais forneceriam serviços a nível de rede. Assim, tais utilitários poderiam fornecer o diretório de máquinas remotas ou mover arquivos entre duas máquinas.

Devido à necessidade de placas virtuais é que a transmissão e recepção de cartas não foi realizada através de um mecanismo de processos. O mecanismo de interrupções seria mais adequado para a metáfora de placa virtual, a ser implementada como um programa residente.

É importante salientar que enquanto o primeiro tipo de utilização seria portável para qualquer implementação, os dois últimos seriam uma solução válida apenas para equipamentos da linha IBM-PC. Em outras máquinas, outros mecanismos deveriam ser disponíveis para permitir a comunicação entre aplicações e a ampliação do sistema operacional (em um ambiente Unix, por exemplo, a solução poderia ser muito mais elegante). Assim, a metáfora de placa virtual deveria ser mapeada de forma dife-

renciada para cada tipo de hardware e/ou sistema operacional.

13 - Detalhes e Perspectivas

A rede aqui descrita foi desenvolvida conforme a metodologia da GRUPPOOTECA. Todo o código fonte foi feito em Modula-2 , não contendo nenhuma instrução em linguagem de montagem ("assembly").

Pesquisas em hipertexto em rede estão sendo desenvolvidas pelo nosso grupo para explorar as perspectivas de se desenvolver uma aplicação que "rode" em rede (tipo número um de utilização).

Agradecimentos

Gostaríamos de agradecer a todos os membros do grupo, sem os quais este trabalho não poderia ser realizado.

Bibliografia

[Itautec88] 'Manual de Desenvolvimento sob SIM/DOS'; Manual Técnico Da Itautec,1988.

[TopSpeed88] 'TopSpeed Modula-2 TechKit'; Jensen & Partners International,1988.

[TopSpeed88] 'TopSpeed Modula-2 User's Manual'; Jensen & Partners International,1988.

[Wirth85] 'Programming In Modula-2'; Niklaus Wirth; 3ª Edição; Berlin, Germany : Springer-Verlag Berlin Heidelberg,1985.

[Messer86] 'Modula-2 : Constructive Program Development'; P. A. Messer e I. Marshall; GB. Blackwell Scientific Publications,1986.

[MEYER88] 'Object-Oriented Software Construction'; Bertrand Meyer; Cambridge, GB. Prentice-Hall Int., 1988.

[Tanenbaum88] 'Computer Networks'; Andrew S. Tanenbaum; 2ª Edição, Amsterdam,The Netherlands : Prentice Hall International,Inc,1988.

[Melgarejo88] 'GRUPPOOTECA : Uma Metodologia de Desenvolvimento de Software'; L. F. B. Melgarejo et all; Centro Tecnológico-CEC-UFSC-EDUGRAF, 1988.

[Laurel90] 'The Art of Human-Computer Interface Design'; Brenda Laurel, Addison-Wesley Publishing Company, Inc., 1990.

[Giozza86] ' Redes Locais de Computadores: Tecnologia e Aplicações'; W. F. Giozza et all; São Paulo: McGraw-Hill,1986.

[Tarouco86] 'Redes de Computadores Locais e de Longa Longa Distância'; Liane Tarouco, McGraw-Hill, 1986.