SUPORTE DE COMUNICAÇÃO PARA DESENVOLVIMENTO DE APLICAÇÕES DISTRIBUÍDAS EM UM AMBIENTE IBM 3090 / PC

Katia Barbosa Saikoski [1]
Eduardo Born Lampert [2]

Pontifícia Universidade Católica do Rio Grande do Sul - PUCRS
Instituto de Informática
Av. Ipiranga, 6681 - Prédio 30 - Cx. postal 1429 - CEP 90819-900 - Porto Alegre - RS
Fone: (051) 339-1511 Ramal 3211 Fax: (051) 339-1564

E-mail: katia@brpucrsm.bitnet

RESUMO

Este artigo apresenta o nível de comunicação de um modelo desenvolvido para criação de aplicações distribuídas no ambiente IBM 3090 / PCs - Projeto COSHE [3]. É analisado o ambiente do projeto e a alternativa encontrada para sua implementação.

São analisadas algumas considerações sobre aplicações distribuídas e apresentadas as funções de comunicação desenvolvidas.

ABSTRACT

This paper presents the communication level from a model developed to create distributed applications in IBM 3090 / PCs environment - COSHE [3] Project. The project environment and the choice for its implementation are analyzed.

Some considerations about distributed application are analyzed and the communications functions are presented.

---

[1] Engenheira Eletricista (UFRGS, 1989), mestranda em Ciência da Computação (UFRGS), professora do Instituto de Informática / PUCRS.

[2] Bacharelando em Informática pela PUCRS.

[3] Este trabalho é parte de um projeto denominado COSHE - COmunicação entre Sistemas HEterogêneos -, financiado pela FAPERGS e com apoio PUCRS e CAPES (Grupo PET).

1. INTRODUÇÃO

A utilização da informática para processar informações tem passado por diversas evoluções. Hoje em dia as redes de computadores estão em todos os ambientes e um dos principais objetivos é compartilhar os recursos existentes. A migração para estes ambientes ocasionou uma mudança nas plataformas de trabalho onde predominava o ambiente de grande porte e foi em alguns casos substituído por um ambiente de microcomputadores ou estações de trabalho.

Neste ponto surge o impasse entre a utilização do ambiente de grande porte e a microinformática. As aplicações desenvolvidas para os computadores de grande porte tenderiam a desaparecer caso não fosse possivel desenvolver uma forma de utilizá-las através de outras máquinas. A simples interligação para compartilhar recursos em uma rede deveria ser renovada para permitir o desenvolvimento de uma aplicação que pudesse ser executada em mais de um ambiente simultaneamente. Neste contexto surge a alternativa de processamento distribuído com o objetivo de permitir que uma aplicação seja desenvolvida para diversos ambientes computacionais e possa ser executada a partir de uma plataforma, mas parte de seu processamento ser executado em outra.

A criação de um ambiente adequado para este tipo de desenvolvimento envolve diversas áreas do conhecimento. São estudos relacionados principalmente a processamento distribuído, redes de computadores, sistemas operacionais e compiladores.

Assim, foi idealizado o Projeto COSHE que tem como objetivo principal o desenvolvimento de um ambiente para projetar aplicações distribuídas. Este projeto está dividido em três áreas de acordo com os conhecimentos envolvidos: comunicação, sistema operacional e aplicação. Este trabalho apresentará o nivel de comunicação.

2. PROJETO COSHE - COmunicação entre Sistemas HEterogêneos

O Projeto COSHE tem como objetivo implementar uma interface de programação em diversas plataformas, disponibilizando funções que serão utilizadas para o desenvolvimento de aplicações distribuídas.

A idéia de desenvolver este projeto surgiu da hipótese de se utilizar o Processador Vetorial existente no IBM 3090 da PUCRS para processar informações provindas de um PC. Assim, uma determinada aplicação que fosse desenvolvida em um PC poderia chamar rotinas desenvolvidas para o Processador Vetorial, disparar processos para executar estas rotinas e receber os resultados. Assim, uma aplicação seria executada em parte no PC e em parte no IBM 3090.

Para que as aplicações possam utilizar os melhores recursos de cada máquina, as funções devem ser implementadas para as diversas plataformas. Em função destas plataformas terem sistemas operacionais diferentes e hardware diferentes, e para tornar o desenvolvimento mais modular, foi necessário estruturar o sistema em três níveis hierárquicos nos mesmos moldes do Modelo de Referência OSI [TAN89] [TAR86], sem ser estar de acordo com este. Os níveis do projeto são: Nível de Comunicação, Nível de Sistema Operacional e Nível de Aplicação, como mostra a figura 2.1 que exemplifica a interligação entre dois ambientes utilizando o modelo proposto.

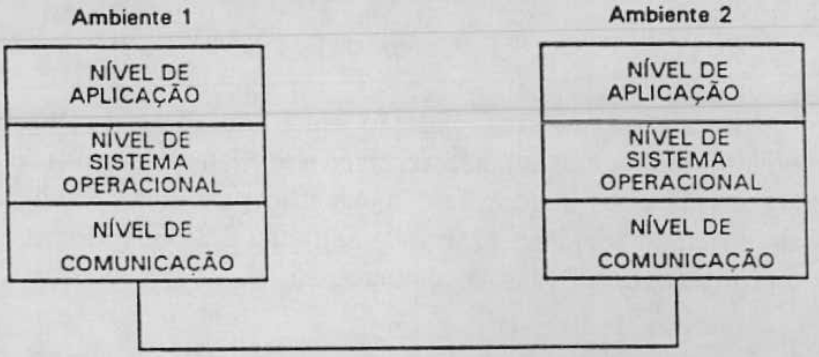

Figura 2.1 - Modelo do Sistema

O nível de aplicação é responsável por desenvolver uma interface para programação neste ambiente. O nível de sistema operacional controla a transferência entre a aplicação através das interfaces de aplicação e o nível de comunicação. Este é responsável por enviar as solicitações ao sistema remoto.

3. DESCRIÇÃO DO AMBIENTE

O ambiente desta fase inicial do desenvolvimento do projeto consistiu de um IBM 3090 e um PC. O objetivo era executar uma aplicação que envolvesse os dois ambientes.

A característica principal do IBM 3090 [IBM90] da PUCRS é sua capacidade de realizar o processamento corporativo da PUCRS e simultaneamente auxiliar nos projetos de pesquisa. Para a área acadêmica é utilizado o sistema operacional MUSIC/SP, que foi a base para desenvolvimento no ambiente do IBM 3090. Optou-se por este sistema operacional (que tem como suporte o sistema operacional VM) pois é o ambiente mais utilizado para desenvolvimento de projetos no Instituto de Informática. No PC, foi utilizado sistema operacional DOS.

A conexão entre os dois sistemas pode ser vista na figura 3.1.

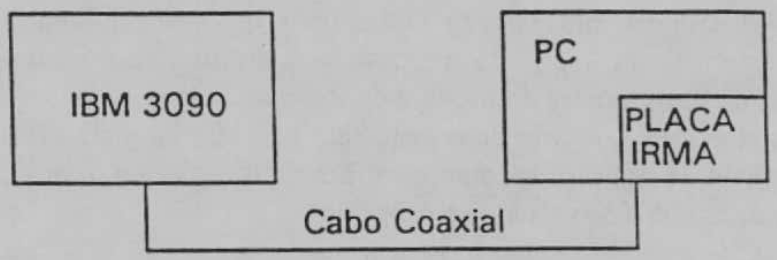

Figura 3.1 - Conexão IBM 3090 - PC

A figura 3.1 mostra a ligação entre um PC e o IBM 3090 através da utilização de uma placa de compatibilização entre eles. Esta placa - Placa IRMA - converte os sinais originados da controladora de comunicação para sinais inteligíveis no PC. Uma visão melhor do sistema poderá ser observada na figura 3.2 onde são apresentados algumas das camadas de software envolvidas na comunicação.

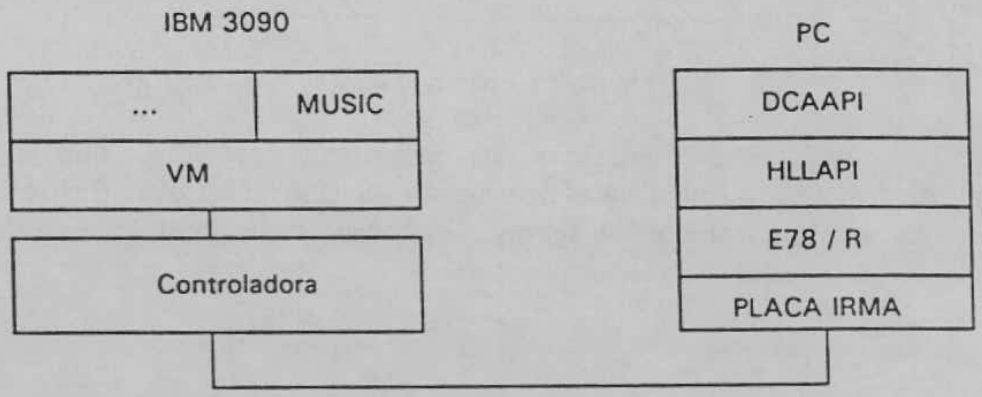

Figura 3.2 - Esquema da Conexão IBM 3090 - PC (Hardware + Software [4])

Para desenvolvimento do nível de comunicação, foi escolhida a alternativa de utilizar a placa de comunicação IRMA (que permite a conexão física entre o PC e o 3090 através da conversão dos sinais da controladora 3274 de forma que sejam inteligíveis ao PC) e camadas de software que implementam funções de acesso à placa. Estas camadas de software são:

- DCA API (Application Program Interface): Interface de baixo nível que permite a comunicação de aplicações desenvolvidas em IBM PC/PS com um computador host;

- DCA HLLAPI (High Level Language Application Program Interface): Funções de alto nível que quando chamadas em uma aplicação criada no PC através da DCA API permitem a comunicação com o host;

[4] [DCA92], [DCA92a], [IRM92]

- LIM (Language Interface Module): A LIM é encarregada da tradução de uma chamada de função HLLAPI criada em uma determinada linguagem de programação de alto nivel (C, Pascal, Cobol, etc.) em chamadas padrão compreendidas pela HLLAPI.

- COMMCHK Hardware Diagnostics: Permite o diagnóstico do hardware de comunicação assim como a operação de SET / RESET da placa IRMA.

4. CONSIDERAÇÕES SOBRE APLICAÇÕES DISTRIBUÍDAS

Alguns aspectos sobre aplicações distribuídas devem ser analisadas para o desenvolvimento do nivel de comunicação. Segundo [ROF 92], aplicação distribuida é um conjunto de funções que são executadas em conjunto ou separado em várias plataformas de hardware e software interligadas. Tais funções podem, além disso, acessar dados igualmente distribuidos por estas plataformas. Assim, para o desenvolvimento de aplicações distribuidas o objetivo é integrar as funções dos ambientes envolvidos.

A relação entre as partes distribuídas de uma aplicação podem ser: troca de informações entre funções, chamada/retorno ou dirigida a eventos. Os três modelos que implementam tais relações são o modelo conversacional, chamada remota de procedimentos e mensagens/enfileiramento.

O modelo conversacional pressupõe que duas funções concordam em quem pode transmitir e quem pode receber, baseado em um protocolo previamente estabelecido. A inversão trasmite/recebe ocorre até que o processamento tenha sido completado e a conversação então é finalizada. Tal modelo pode ser implementado através de uma interface de programação (API - application program interface).

O modelo de chamada de procedimentos remotos implica em uma requisição de serviço, a execução de uma tarefa até a finalização do processo. Os serviços podem ser locais ou remotos, introduzindo o conceito de transparência em que a aplicação não necessita identificar onde o serviço está sendo realizado.

No modelo mensagem/enfileiramento, a comunicação entre as funções é através da inclusão de uma mensagem de um evento na fila do sistema, que é direcionada para a fila da função chamada. A mensagem é retirada da fila e então processada.

Os modelos acima podem ser utilizados em diferentes partes de uma aplicação distribuida. Uma função pode ser cliente de outra que pode estar se comunicando através de uma interface de aplicação. Para a etapa do projeto em questão, foi necessário desenvolver uma interface própria e utilizar alguns serviços de rede, além de um modelo de chamada de procedimentos remotos.

5. IMPLEMENTAÇÃO DAS FUNÇÕES

A identificação das funções do nível de comunicação do modelo criado foi em função de uma análise dos sistemas operacionais VM , MUSIC e DOS e a criação de uma aplicação distribuída com o objetivo de identificar a execução e comunicação entre processos nestes sistemas.

O nível de comunicação é responsável por transmitir e receber dados entre o host e o PC. As diversas camadas de software utilizadas (DCAAPI, HLLAPI, LIM) forneceram a estrutura de dados e algumas funções de programação. A comunicação entre o host e o PC é feita através do chamado Espaço de Apresentação (PS - Presentation Space) que disponibiliza um buffer que representa os dados da tela. Assim, quando se quer transmitir de um ambiente para outro, o espaço de apresentação deve ser preenchido, isto é, a tela deve ser preenchida de um lado para ser recebida do outro lado. A manipulação do espaço de apresentação é feita através das funções HLLAPI que permanece residente para atender as chamadas do nível de comunicação.

O nível de comunicação é implementado com transparência funcional, ou seja, o usuário não percebe que está sendo feita a comunicação com o Host. Além disto, os serviços devem ter as mesmas características tanto no PC como no Host, o que é chamado de equivalência funcional. A comunicação ocorre de tal forma que vários processos se comunicam entre host e PC de forma simultânea e exclusiva.

No lado do PC, as funções foram desenvolvidas em linguagem C. São funções que fazem chamada às funções da HLLAPI (que fica residente) para efetuar a comunicação. A cada chamada de função são realizados os seguintes passos:

a) São passados os seguintes parâmetros:

* número da função da HLLAPI requisitada - **int ehl_func**;
* string de dados - **char ehl_data [256]**;
* tamanho da string de dados - **int ehl_len**;
* código de retorno - **int ehl_retc**;

A chamada da função fica:

```
tclim ( &ehl_func ; &ehl_data ; &ehl_len ; &ehl_rect )
```

b) A LIM interpreta as requisições e constrói um Bloco de Controle (PCB), este bloco é passado para a HLLAPI através da interrupção X7F (não utilizada pelo DOS).

c) O módulo HLLAPI interpreta a chamada de função passando-a para o espaço de apresentação onde a tarefa é executada. Posteriormente retorna os dados requisitados e um código (ehl_rect) que indica o status da execução.

d) O controle retorna a LIM, que aguardava o retorno da interrupção X7F, retornando os valores em forma de parâmetros ao programa solicitante (nível de comunicação) que retoma o controle.

No lado do Host IBM 3090 o nível de comunicação foi desenvolvido e testado utilizando o sistema operacional MUSIC/SP. A codificação foi feita em linguagem REXX que adequa à necessidade do sistema ser multi programado. O nível de comunicação no host executa as seguintes tarefas: abertura de um arquivo para receber os dados do espaço de apresentação, recebimento dos blocos transmitidos, fechamento do arquivo e notificação de erros. As informações são passadas para o host sob forma de um arquivo texto que é transmitido linha a linha da tela através do espaço de apresentação utilizando a função **SendKey** da HLLAPI.

Para manter a comunicação no PC, que não é multi programado, optou-se por manter alocado um espaço de memória para a execução de troca de dados. Isto é feito através de uma rotina de comunicação de tipo TSR (Terminate and Stay Ready), podendo, desta forma, ser acessada pelos programas em execução.

Por fim, existe uma biblioteca de rotinas, que será utilizada pelos programas de aplicação para desenvolvimento da execução e comunicação entre processos.

As funções desenvolvidas para o nível de comunicação no PC foram:

FUNÇÃO	DESCRIÇÃO DA FUNÇÃO
reset	utilizada para reinicializar a HLLAPI
set_session_param	ajusta os parâmetros da sessão com o host
query_sessions	retorna os parâmetros da sessão
connect_ps	conecta o programa ao espaço de apresentação
sendkey	transmite strings e teclas de controle para o host
smartwait	utilizada para temporização da transmissão
copy_ps	copia o espaço de apresentação para o programa
conecta	estabelece a sessão com o host
desconecta	realiza a desconexão
envia_texto	envia blocos de dados para o host
recebe_texto	recebe blocos de dados do host

As funções no IBM 3090 foram desenvolvidas em REXX em um único módulo. Este inclui as funcionalidades equivalentes às funcionalidades implementadas no PC (recebimento/envio de blocos, tratamento do espaço de apresentação e de arquivos e tratamento das requisições).

Para utilizar as funções disponíveis no PC, todos os módulos de software do nível de comunicação devem ser executados. As funções ficam disponíveis através de um programa residente que se liga com a aplicação do usuário permitindo a troca de dados. Esta

camada de software faz a conversão dos dados e a chamada do módulo de comunicação e transfere o controle da execução para a função especificada. O resultado é devolvido à aplicação que continua sua execução.

6. TRATAMENTO DE EVENTOS

Com a configuração de hardware e software apresentadas, o PC se torna um terminal do IBM 3090 e a transmissão entre eles é feita através do espaço de apresentação. Para que o desenvolvimento de uma aplicação distribuída, foram construídas algumas funções que são transmitidas entre IBM 3090 e PC através do espaço de apresentação. Um dos itens importantes na construção destas funções foi a análise detalhada do funcionamento dos terminais 3270 a nível de temporização. Assim, foram criadas rotinas de tratamento de eventos (tais como o XCLOCK) e espera de eventos. Com estas rotinas pode-se testar a solicitação e execução das requisições entre os sistemas.

O XCLOCK é um sinal do host (IBM 3090) indicando que está ocupado. Para que possam ser enviadas novas seqüências de teclas, uma aplicação deve esperar que o evento XCLOCK termine. Usualmente, o XCLOCK ocorre sempre após o envio de uma tecla AID (attention identifier) - no PC esta tecla corresponde ao ENTER.

Após o envio de uma tecla AID, o XCLOCK é ativado. Durante o tempo de ativação, o sinal XCLOCK não é detectado e não se pode enviar mais teclas. Isto poderá causar alguns problemas de reconhecimento do estado do XCLOCK. O comportamento do XCLOCK pode ser observado na figura 6.1.

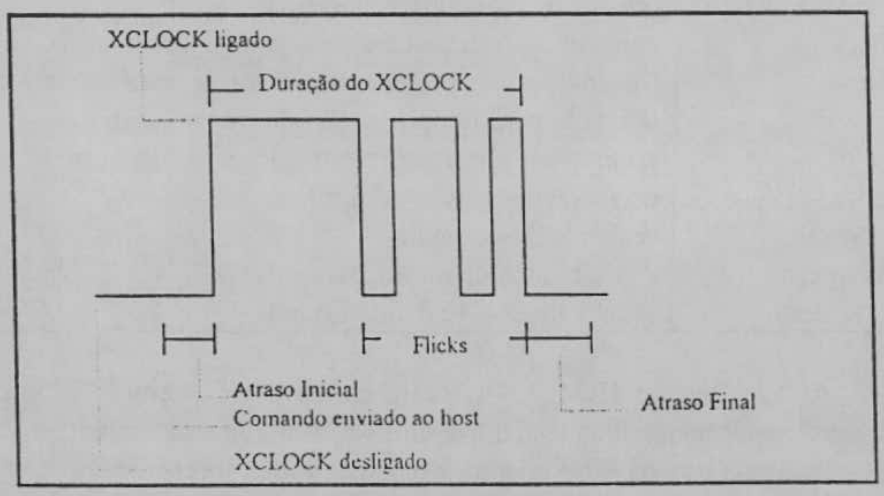

Figura 6.1 - Seqüência de tempo do XCLOCK

Tornou-se necessário a criação de uma rotina para tratamento de espera de eventos. Esta rotina pode ser chamada quando um processo é disparado no host para notificar seu término. isto se deve ao fato de que quando um processo é disparado, o XCLOCK é ativado e após sua desativação, o processo estará ainda sendo executado. A rotina de tratamento de espera faz o controle da execução do processo para que novas teclas sejam enviadas após seu término.

7. TESTES DO SISTEMA

Os níveis do modelo foram desenvolvidos separadamente. Assim, para testes do módulo de comunicação, desenvolveu-se uma aplicação de transferência de arquivos. Assim, o sistema funciona da seguinte forma: a aplicação no PC é chamada; ela contém uma função de transferir arquivos que se caracteriza por ser remota; isto faz com que uma sessão entre um PC e o host seja aberta de forma transparente para o usuário; uma vez aberta a sessão, o arquivo é copiado para o espaço de apresentação, que por sua vez é copiado para um arquivo no host.

Da mesma forma que transferência de arquivos, qualquer solicitação de execução funcionaria da seguinte forma: a função remota é chamada em uma aplicação; uma sessão é estabelecida entre host e PC; a requisição é passada do PC ao host através do espaço de apresentação; no host o conteúdo do espaço de apresentação é transformado em um comando, executado e o resultado volta ao PC.

Quando a solicitação parte do host para ser executada no PC, o procedimento deve ser diferente. O PC deverá estar com a sessão aberta com o host (denominou-se o PC como servidor de aplicações) para que reconheça a solicitação que chegou. Os testes neste módulo foram feitos através de comandos no host que solicitam tarefas de disco no PC (comandos DOSDIR, DOSCD, DOSREAD, DOSWRITE). Estes comandos são solicitados no host, colocados no espaço de apresentação e reconhecidos no PC através de um programa residente.

Após a integração dos módulos, será disponibilizado uma conta no host para atender às solicitações do PC. Toda aplicação deverá ser executada através de um PC que contenha a configuração mostrada na figura 3.2 mais a camada de software relativa ao modelo proposto.

8. CONSIDERAÇÕES FINAIS

Este artigo apresentou uma alternativa encontrada para criar um ambiente de desenvolvimento de aplicações distribuídas. O projeto COSHE que está em andamento na PUCRS tem como meta implementar este modelo em diferentes plataformas. A justificativa para escolher a plataforma IBM 3090 / PC para início do trabalho é em função da grande utilização dos dois ambientes e suas características específicas.

Para implantar o sistema em outras plataformas, devem ser considerados alguns padrões do tipo RPC (Remote Procedure Call) que permitem maior portabilidade e compatibilidade a nível de implementação. Nenhum padrão foi adotado para esta fase inicial devido às caracteristicas especiais dos sistemas em questão.

A integração entre os diversos módulos não foi executada. As funções que sendo desenvolvidas para atender às necessidades dos outros níveis do modelo incluem tratamento das aplicações, tratamento de sincronismo e de requisições que não foram tratadas a nível de comunicação.

Com estas considerações espera-se ter justificado o desenvolvimento do projeto, sua metodologia e alternativas escolhidas para implementação e testes. Acredita-se que ao final desta etapa do projeto, será possível utilizar os dois sistemas para executar uma aplicação, por exemplo, de computação gráfica onde o PC requisita o processador vetorial para execução de determinados módulos do programa, caracterizando-se então uma aplicação distribuída.

AGRADECIMENTOS

O trabalho apresentado está sendo desenvolvido na PUCRS com apoio FAPERGS que fornece bolsas de iniciação científica e apoio CAPES com um bolsista PET, além de alunos de trabalho de conclusão do curso de Informática.

Os alunos Marcelo Braga Vaz, Rogério Melnick, Rogério Lopes e Alexandre Franco merecem um agradecimento especial por participarem ativamente na especificação e implementação do desenvolvimento do projeto.

Os autores gostariam de agradecer ao professor Avelino Zorzo por sua experiência e participação.

REFERÊNCIAS BIBLIOGRÁFICAS

[DCA92] DCA. **API Support Information.** DCA Digital Communications Associate, 1992.

[DCA92a] DCA. **E78 Plus and E78 Lite.** DCA Digital Communications Associate, 1992.

[IBM90] IBM. **IBM Systems and Products Guide**, 1990

[IRM92] DCA. **IRMA3: User Guide.** DCA Digital Communications Associate, 1992.

[ROF92] ROFRANO, J.J. Jr. **Design Considerations for Distributed Applications**. IBM Systems Journal, v.31, n.7, p.564-589, 1992.

[TAN89] TANEMBAUM, Andrew. **Computer Networks**. Prentice-Hall International, Inc., 2ed. 1989.

[TAR86] TAROUCO, Liane M. R. **Redes de Computadores - Locais e de Longa Distância**. São Paulo: McGraw-Hill, 1986.