E.B.Lucena*, P.R.F.Cunha** e J.A.S.Monteiro***

SUMÁRIO

Este trabalho tem o objetivo de descrever o desenvolvimento de uma solução modular para a programação da Estação de Transporte da Rede CEPINNE, bem como o mapeamento desta modularização para o ambiente de programação sequencial disponível na nossa universidade. As alternativas de integração deste software ao ambiente hospedeiro, bem como a opção escolhida são também apresentadas.

ABSTRACT

This work has the objective of describing the development of a modular solution for the programming of the CEPINNE Network Transport Station and the mapping between this modularization and the programming environment available in our university. The alternatives how to integrate this software to the host computer and the implemented option are also presented.

* Bacharel em Ciência da Computação (UFPE,1984), Sistemas Operacionais, Redes de Computadores, Sistemas Distribuídos, REDIS/UFPE, Av. Prof. Luiz Freire s/n - 50.000 - Recife - PE - (081) 271.2510

** Engenheiro Eletricista (UFPE,1974), Mestre em Informática (UFPE,1977), Ph.D. em Computação (Waterloo, Canadá,1981). Professor Adjunto do Deptº de Informática da UFPE e membro do Grupo de Redes de Computadores e Sistemas Distribuídos da UFPE. Metodologias de Programação, Sistemas Distribuídos, Redes de Computadores.

*** Engenheiro Eletricista (UFPE,1979), Mestre em Engenharia Elétrica (USP,1982), Professor Assistente do Deptº de Informática UFPE, Redes de Computadores, Especificação e Validação de Protocolos, Sistemas Distribuídos.

1. INTRODUÇÃO

São várias as razões que nos levam a interligar os diversos sistemas computacionais existentes. Dentre elas podemos citar a necessidade de compartilharmos de recursos (informações, potencial computacional, dispositivos, etc.), a necessidade de desenvolvimento de recursos humanos especializados, a necessidade de um meio de comunicação mais poderoso, confiável e barato, etc.

Foi com estes e outros objetivos em mente que a SEI (Secretaria Especial de Informática) elaborou, em janeiro de 1981, o Projeto CEPINNE (Centro Piloto de Serviços Públicos de Teleinformática para Aplicações em Ciência e Tecnologia - Região Norte - Nordeste) [9], o qual envolveria a interligação dos diversos recursos computacionais disponíveis nas universidades das regiões Norte e Nordeste.

A participação da UFPE (sob o comando de seu Grupo de Redes de Computadores e Sistemas Distribuídos - REDIS/UFPE) neste projeto passou a ser mais efetiva a partir de janeiro de 1984, quando da celebração de um convênio entre UFPE, UFPB, EMBRATEL e FUNAPE (Fundação de Apoio a Pesquisa e Extensão da Universidade Federal da Paraíba), que visava ao desenvolvimento, implementação e testes de um Protocolo de Transportes para a Rede CEPINNE.

Este artigo tem o objetivo de descrever o desenvolvimento de uma solução modular para a programação da nossa Estação de Transporte, bem como o mapeamento deste modularização para o ambiente de programação, desprovido de paralelismo, disponível na UFPE. Na seção 2, descrevemos, sucintamente, o Protocolo de Transporte adotado. A seção 3 apresenta a estrutura modular do software (em termos de processos e monitores) da nossa Estação de Transporte e o mapeamento desta estrutura para o nosso ambiente. As alternativas de integração deste software ao ambiente hospedeiro, bem como a opção escolhida são apresentadas na seção 4. Finalmente, na seção 5 tecemos algumas considerações finais e identificamos futuros trabalhos que darão continuidade a este projeto.

2. PROTOCOLO DE TRANSPORTE ADOTADO PARA A REDE CEPINNE

Sabe-se que um dos fatores mais importantes no projeto de uma Rede de computadores é a forma com que os sistemas abertos comunicantes conversarão de modo a prover uma comunicação confiável, eficiente e sincronizada. A fim de reduzir a complexidade do projeto e tendo em mente a padronização da interconexão de sistemas abertos, a ISO ("International Organization for Standardization") elaborou o Modelo de Referência OSI (Open Systems Interconnection), o qual estrutura as redes em sete níveis ou camadas, com funções bem definidas, onde cada camada utiliza-se diretamente dos serviços fornecidos pela camada imediatamente inferior [11]. As regras e convenções segundo as quais uma entidade de nível N "conversa" com outra entidade de mesmo nível são globalmente conhecidas como Protocolo de Nível N.

Nesta arquitetura, o Protocolo de Transporte, o objeto de nosso trabalho, corres-

ponde ao protocolo de nível 4. O principal objetivo deste nível é prover uma transferência fim a fim confiável e eficiente. Tal transferência deverá envolver, eventualmente, o fornecimento de serviços tais como multiplexação de circuitos virtuais, quebra e remontagem de mensagens, controle de fluxo, controle de erros fim a fim, recuperação de falhas no nível de Rede (nível 3 na arquitetura do Modelo OSI) — RESET's e RESTART's — , etc. Podemos dizer, em outras palavras, que a camada de transporte procura preencher o eventual espaço existente entre o nível de qualidade de serviço requisitado pelo nível de Sessão (nível 5 na arquitetura do Modelo OSI) e os serviços efetivamente oferecidos pelo nível de Rede.

Quanto ao protocolo de Transporte para a Rede CEPINNE, estabeleceu-se, por questão de tempo, adotar como referência uma especificação preliminar de um protocolo de Transporte desenvolvido pela PUC/RJ [5]. Esta especificação foi revisada, por membros da UFPE, UFPB e PUC/RJ, quando da reunião do LARC (Laboratório Nacional de Redes de Computadores), ocorrida em abril de 1984, sendo efetuados alguns esclarecimentos, retificações e simplificações. A partir destas discussões foram elaborados dois documentos: a especificação dos serviços providos e requeridos por este protocolo de Transporte [1] e a especificação deste protocolo propriamente dito [8]. Decidiu-se também, com o intuito de agilizar a implementação, por não se implementar, ao menos inicialmente, serviços opcionais que o protocolo se propõe a prover, tais como multiplexação, recuperação de falhas, controle de erros fim a fim, transferência não orientada a conexão, delegando estas funções aos níveis superiores.

Este protocolo de Transporte é baseado na troca de fragmentos (unidades de informação de controle e trechos de mensagens), os quais são divididos em dois grupos: estabelecimento e encerramento de conexões (= ligações) e transferência de dados. São eles:

ESTABELECIMENTO E ENCERRAMENTO DE CONEXÕES

- PCONEX (Pedido de Conexão)
- CONACEITA (Aceitação de Conexão Solicitada)
- REJCON (Rejeição de Conexão)
- ACKAC (Confirmação de Aceitação de Conexão)
- FCONEX (Encerramento de Conexão)
- ACKFE (Confirmação de Encerramento de Conexão)

TRANSFERÊNCIA DE DADOS

- DADOS (Transporte de Dados Normais)
- INTERRUPÇÃO (Transporte de Interrupção)
- TELEGRAMA (Transporte de Telegramas)
- ACKDADOS (Confirmação de Recepção de Dados e/ou Interrupção)

As conexões deste protocolo permitem uma transferência full-duplex, ou melhor, o fluxo de dados pode ocorrer nos dois sentidos simultaneamente, e o encerramento da conexão

num sentido não implica necessariamente no encerramento no sentido oposto (a menos do aborto da conexão). As temporizações envolvidas neste protocolo são basicamente duas: a temporização de retransmissão de fragmentos e a temporização de inatividade remota (tempo máximo, determinado pelo usuário, que a Estação de Transporte suportará sem receber nenhum fragmento da entidade remota). O serviço requerido, por este protocolo, do nível inferior é o serviço de Circuitos Virtuais do nível 3 da Recomendação X.25 [3]. Maiores detalhes acerca deste protocolo e seus serviços podem ser encontrados nas referências [8] e [1], respectivamente.

3. ESTRUTURA MODULAR DO SOFTWARE DA ESTAÇÃO DE TRANSPORTE

Procuramos estruturar o software da Estação de Transporte em módulos de maneira que pudéssemos atacar o problema por etapas, o que consequentemente facilitaria a implementação e os testes desta Estação.

Num ambiente apropriado, seria possível também a exploração de todos os pontos de paralelismo existentes em um software desta natureza. Por exemplo, o tratamento da chegada de primitivas vindas do(s) nível(eis) superior(es) poderia, desde que haja algum mecanismo de sincronização do acesso às variáveis comuns, ser feito em paralelo com o tratamento da chegada de primitivas vindas do nível inferior e com o tratamento das indicações de estouro de temporizadores.

De um modo geral, esta estrutura derivou, em certos aspectos, da metodologia de programação proposta em [10]. Esta metodologia se baseia numa programação por eventos, sendo ocorrência destes eventos representada pela presença de dados. No nosso caso, a ocorrência de eventos é indicada pela presença de primitivas nas filas dos Monitores de Interface. Tais monitores podem ser controlados por um gerenciador de interfaces como em [10]. A ocorrência destes eventos (presença de primitivas na filas de recepção) irá disparar a execução de um certo conjunto de ações (consumo da indicação de ocorrência do evento, envio de novas primitivas, disparo de temporizações, atualização de variáveis, etc.).

A estrutura do software da Estação de Transporte é apresentada no diagrama da figura 3.1, no qual os arcos direcionados indicam o sentido do fluxo de dados, os módulos entre circuitos simbolizam Monitores (entidades passivas) e os módulos entre retângulos simbolizam Processos (entidades ativas). A comunicação entre cada um dos processos é feita com o auxílio de dois monitores de interface (que contêm as filas de recepção e transmissão de primitivas e podem ser controlados por um gerenciador de interfaces). A seguir apresentaremos a estrutura e, em seguida, descreveremos cada um dos seus módulos.

PROCESSOS

Pi:

Representam os diversos processos que usufruem dos serviços providos pela Estação de Transporte.

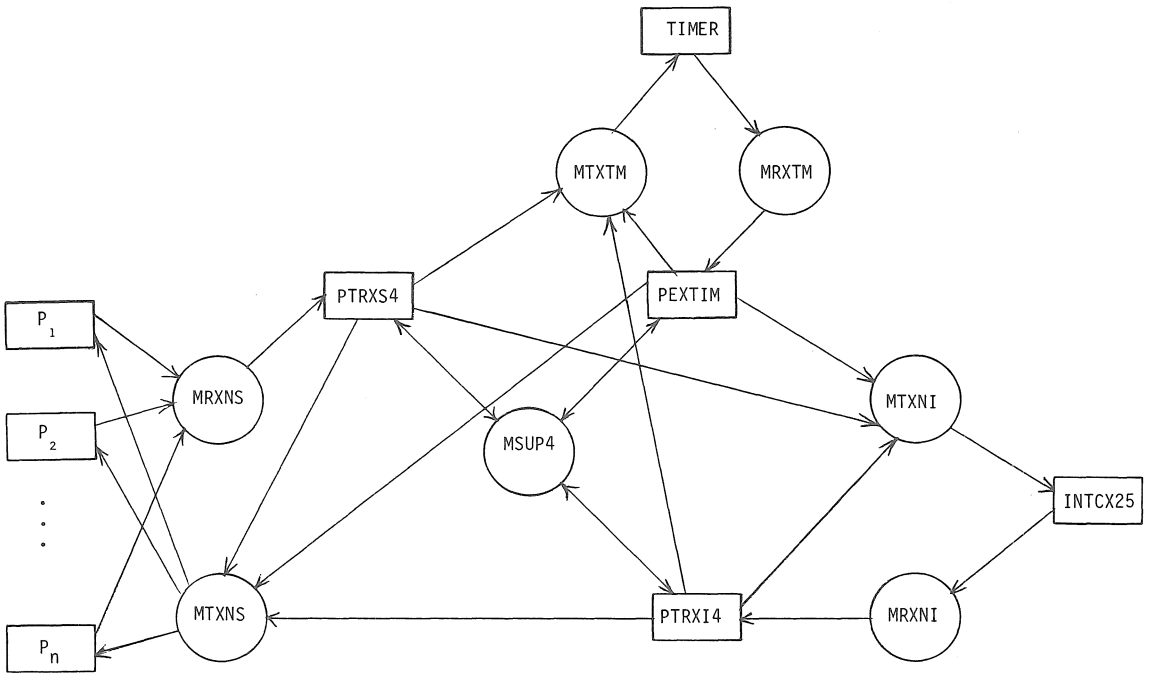

FIG. 3.1 - ESTRUTRURA DO SOFTWARE DA ESTAÇÃO DE TRANSPORTE

TIMER:

Processo responsável pelas temporizações do sistema. Não é propriamente um módulo da Estação de Transporte. Guarda as solicitações de temporizações (identificação do processo solicitante, identificação do tipo de temporização e conexão, tempo solicitado), computa este tempo solicitado e notifica o processo solicitante quando da ocorrência de time-out (através do envio de uma indicação sob a forma de primitiva). Pode ser utilizado simultaneamente pela Estação de Transporte e qualquer outro processo usuário que dele necessitar.

PEXTIM:

Trata da ocorrência do evento "estouro de temporizador" (evidenciado pela presença de uma indicação na fila de recepção) na Estação de Transporte. Dentre as ações contidas no conjunto de tratamento para este evento podem ser citadas o consumo da indicação, solicitações de novas temporizações, atualização do bloco de contexto da ligação, envio de primitivas aos níveis superior e/ou inferior, etc.

PTRXS4:

Processo responsável pela execução de ações pela ocorrência de eventos representados pela presença de primitivas vindas do(s) nível(eis) superior(es), tais como ABRECON, ESPERACON, FECHACON, ENVMSG, ENVINT, etc. Podemos citar ações como consumo da indicação, solicitação de portas, blocos de controle de ligação, envio de primitivas de solicitação de circuito virtual (ESTABCV), etc.

PTRXT4:

Processo responsável pela execução de ações em virtude da ocorrência de eventos representados pela presença de primitivas originadas do nível inferior (no nosso caso, a interface com o conversor CX.25). Tais ações incluem o consumo de indicação, o envio de primitivas que sinalizem ao nível superior e estabelecimento de conexões, a ocorrência de abortos de conexões, a chegada de mensagens, etc.

INTCX25:

Este processo efetua a interface entre a Estação de Transporte e o conversor CX.25.. Evidentemente não pode ser considerado efetivamente como um módulo da Estação de Transporte. Suas tarefas englobam o tratamento de primitivas vindas da Estação de Transporte (ex. ESTABCV, LIMPACV, TRPACOTE, etc.), o envio de primitivas de indicação ou resposta à Estação de Transporte (IND-RESET, RESP-ESTABCV, etc.), a interpretação e tratamento dos comandos vindos do conversor CX.25, etc. Maiores detalhes sobre esta interface podem ser obtidas na referência [6].

MONITORES

MRXNS:

Monitor de recepção de primitivas originadas do(s) nível(eis) superior(es)

(ABRECON, FECHACON, ABORTACON, etc.). Contêm a fila de recepção de primitivas do(s) nível(eis) superior(es). As operações permitidas são inserção de primitiva no final da fila e recuperação da primeira primitiva da fila.

MRXNI:

Monitor de recepção de primitivas originadas do nível inferior (INTCX25). Contêm a fila de recepção de primitivas enviadas pela interface com o conversor CX.25. (IND-ESTABCV, RES-TRPACOTE, etc.). As operações permitidas são as mesmas do MRRXNS.

MRXTM:

Monitor de recepção de indicações de time-out (primitiva ESTOURO-TEMPORIZADOR). Contêm a fila de notificações de time-out. As operações permitidas são as mesmas do MRXNS.

MTXNS:

Monitor de transmissão de primitivas destinadas ao(s) nível(eis) superior(es). Contêm a fila de transmissão de primitivas ao(s) nível(eis) superior(es) (CHEG-MENSAGEM, IND-ABORTO, RESP-ABRECON, etc.). As operações permitidas são as mesmas.

MTXNI:

Monitor de transmissão de primitivas ao nível inferior. Contêm a fila de transmissão de primitivas destinadas ao nível inferior (ESTABCV, TRPACOTE, LIMPACV, etc.). As operações permitidas são as mesmas.

MTXTM:

Monitor de transmissão de solicitações de temporizações ao Temporizador (TIMER). Contêm a fila de solicitações de temporizações (LIGA-TEMPORIZADOR). As operações permitidas são as mesmas.

MSUP4:

Monitor de supervisão da Estação de Transporte. Contêm as estruturas de controle da Estação de Transporte (Mapas de Portas, Mapas de Circuitos Virtuais, Blocos de Controle das Ligações, Blocos de Espera de Ligações, etc.). As operações permitidas são basicamente consulta e atualização.

3.1. MAPEAMENTO PARA O AMBIENTE COMPUTACIONAL DA UFPE

O sistema de computação hospedeiro disponível na UFPE é o DECsystem-10 (com Sistema Operacional em regime de Time-Sharing, versão 7.01A). O fato deste ambiente não nos permitir implementar facilmente regiões críticas controladas por mecanismos de sincronização tipo Monitores (uma vez que não dispomos no momento de nenhuma implementação de linguagem concorrente), praticamente inviabilizou a implementação da estrutura apresentada anteriormente. Desta forma, a maneira mais conveniente de permitir a comunicação entre os processos encontrada foi utilizar o software utilitário disponível no TOPS-10, IPCF ("Inter-Process Communication Facility") [4]. Com o uso deste utilitário, os processos se comunicam através da troca de mensagens ou pacotes IPCF. A cada processo estão asso-

ciados uma fila de transmissão e outra de recepção de pacotes. O gerenciamento destas filas é praticamente feito pelo TOPS-10. Os nossos monitores de transmissão e recepção (monitores de interface) foram mapeados para as filas de transmissão e recepção de pacotes IPCF. Os processos PTRXI4, PTRXS4 e PEXTIM que, na estrutura, concorriam ao acesso das mesmas informações (Bloco de Controle das Ligações, Mapas de portas, etc.) foram mapeados para um único processo. Evidentemente esta solução não é a mais eficiente e estruturada, entretanto foi a mais imediata que encontramos para o curto prazo de conclusão do projeto que dispúnhamos.

4. ALTERNATIVAS DE INTEGRAÇÃO E OPÇÃO ADOTADA

Existem basicamente três maneiras de se integrar a implementação de um protocolo no contexto de um Sistema Operacional hospedeiro [7]. A primeira é implementar o protocolo sob a forma de um ou mais processos usuários (abstração utilizada pela maioria dos Sistemas Operacionais a fim de prover um ambiente de execução para os programas dos usuários). A segunda forma é incorporar a implementação ao Kernel do Sistema Operacional. A última forma é implementar o protocolo fora da máquina hospedeira, em um processador de comunicação ou Front-end. A seguir idenficificaremos as vantagens, desvantagens e requisitos de cada opção.

A implementação do protocolo sob a forma de processo(s) usuário(os) é possivelmente a forma mais simples e a que exige menos alterações no ambinete hospedeiro, uma vez que não é necessário nenhuma modificação no Kernel do Sistema Operacional e são necessários apenas conhecimentos superficiais com respeito ao Kernel, haja vista que a implementação funcionará, em linhas gerais, como apenas mais um usuário do Sistema Operacional. Entretanto, esta opção é provavelmente a menos eficiente em termos de performance, pois o escalonamento de processos em um Sistema Operacional é uma fonte de atraso significativa.

Implementar o protocolo como parte do Kernel do Sistema Operacional possivelmente aliviará o overhead de escalonamento de processos e possivelmente facilitará o projeto de aplicações distribuídas, uma vez que poderíamos inclusive possibilitar a leitura e gravação de informações de/para a rede por processos usuários diretamente através de chamadas ao Sistema Operacional. Todavia esta alternativa possui várias desvantagens, entre as quais a necessidade de profundo conhecimento sobre o Sistema Operacional, o comprometimento da performance do Sistema Operacional como um todo e a dificuldade de manutenção do Kernel e da implementação do protocolo quando da ocorrência de mudanças (modificações de versão do Sistema Operacional, alterações no protocolo, entre outras).

A utilização de um processador de Front-end dedicado também possui suas vantagens e desvantagens. Entre as vantagens, isola o protocolo do Kernel do Sistema Operacional (consequentemente simplificando a implementação e manutenção do mesmo), é mais eficiente (uma vez que praticamente torna o hospedeiro livre de tarefa de comunicação), é mais versátil (haja vista que, com algumas modificações, pode ser utilizada por diferentes máqui

nas hospedeiras). Dentre as principais desvantagens, podemos citar o custo adicional do processador dedicado, a necessidade de uma comunicação, por mais simples que seja, entre a máquina hospedeira e o processador front-end, a dificuldade de interação com os níveis de protocolos adjacentes, etc.

Face ao exposto acima, resolvemos optar por uma alternativa que, poderíamos dizer, combina as características das duas primeiras alternativas. Em outras palavras, nossa implementação foi feita sob a forma de processos, mas que são inicializados juntamente com os processos hibernantes do Sistema Operacional TOPS-10, com certo nível alto de prioridade (mecanismo disponível no ambiente do TOPS-10). Dentre os fatores que nos levaram a tal opção, podemos citar a necessidade de termos, o mais rápido possível, instalada a nossa implementação da Estação de Transporte (o que descartaria a implementação num Front-end ou incorporação direta ao Kernel), a relativa simplicidade de implementação e manutenção, a necessidade de uma melhor interação entre as camadas adjacentes da arquitetura (uma vez que os serviços do nível de Rede já seriam fornecidos por um conversor X.25, o que já implicaria no desenvolvimento de uma interface) e a preferência por não provocar consequências (perda de performance do Sistema Operacional) em toda uma comunidade usuária (o que praticamente inviabilizaria a incorporação ao Kernel).

5. CONSIDERAÇÕES FINAIS E DIRECIONAMENTOS FUTUROS

Esta Estação de Transporte encontra-se atualmente implementada em Pascal seqüencial e testada localmente no DEC-10 da UFPE [2]. Estamos atualmente iniciando o desenvolvimento de uma metodologia de testes para protocolos de comunicação que poderá ajudar de uma maneira mais eficaz os testes integrados UFPE/UFPB do Nível de Transporte da Rede CEPINNE. Estamos na expectativa da instalação do nó REXPAC e no momento, do fornecimento dos conversores CX.25 às universidades envolvidas para que seja possível esta integração. Para dar continuação aos trabalhos relacionados com este projeto, prevemos a inclusão de novos serviços na nossa implementação, tais como multiplexação, recuperação de falhas, controle de erros fim a fim, etc. Também está prevista a especificação do protocolo e seus serviços utilizando-se técnicas mais formais que permitam uma validação mais efetiva deste protocolo.

AGRADECIMENTOS

Gostaríamos de agradecer o apoio financeiro concedido pela EMBRATEL e pela SEI, bem como o apoio técnico fornecido pelo Núcleo de Processamento de Dados da UFPE.

REFERÊNCIAS

[1] CUNHA,P.R.F.; MONTEIRO,J.A.S.; LUCENA,E.B. - "Especificação dos serviços do Protocolo de Transporte para a Rede CEPINNE"; Relatório Técnico submetido à EMBRATEL, dezembro/1984.

[2] CUNHA,P.R.F.; MONTEIRO,J.A.S.; LUCENA,E.B. - "Implementação de um Protocolo de Transporte para a Rede CEPINNE"; Anais do 3º Simpósio Brasileiro sobre Redes de

Computadores, Rio de Janeiro/RJ, abril/1985.

[3] CCITT - Comitê Consultatif Internacional Télégraphique et Téléphonique - "Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipament for Terminals Operating in the Packet Mode on Public Data Networks"; Recommendation X.25, Genebra, novembro/1980.

[4] DIGITAL EQUIPMENT CORPORATION - "Communication Between Processes: IPCF"; Software Netbook 4, julho/1982.

[5] MENASCÉ,D.A. et al. - Especificação Preliminar de um Protocolo de Transporte para Redes Públicas que usam Circuitos Virtuais"; Relatório Técnico, Deptº de Informática, PUC/RJ (Contrato C.GCD-004/80), dezembro/1981.

[6] MONTEIRO,J.A.S.; CUNHA,P.R.F.; RODRIGUES,J.S.F. - "Uso de um Conversor 'START/STOP' - X.25 para o Acesso de Computadores a REDES PÚBLICAS"; Anais do 3º Simpósio Brasileiro sobre Redes de Computadores, Rio de Janeiro/RJ, abril/1985.

[7] NBS - National Bureau of Standards - "Specification of a Transport Protocol for Computer Communications, Volume 5: Guidance for the Implementator"; Institute for Computer Sciences and Technology, U.S. Department of Commerce, janeiro/1983.

[8] SAUVÉ,J.P. et al. - "Especificação do Protocolo de Transporte para a Rede CEPINNE"; Relatório Técnico submetido à EMBRATEL, dezembro/1984.

[9] SEI - Secretaria Especial de Informática - "Projeto CEPINNE"; janeiro/1981.

[10] QUEIROZ,R.J.G.B. - "Uma Metodologia de Programação para Implementação de Protocolos"; Tese de Mestrado, Departamento de Informática, Universidade Federal de Pernambuco/PE, agosto/1984.

[11] ZIMMERMAN,H.; DAY,J.D. - "The OSI Reference Model"; Proceedings of the IEEE, volume 71, dezembro/1983.