UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PROTOCOLO DE TRANSPORTE PARA APOIAR A INTERCONEXÃO COM COMPUTADORES DE GRANDE PORTE

por

RICARDO RODRIGUES BRANCO

Dissertação submetida como requisito parcial para
a obtenção do grau de Mestre em
Ciência da Computação

Profª Liane Margarida Rockenbach Tarouco
Orientador

Porto Alegre, março de 1986

Branco, Ricardo Rodrigues

Protocolo de transporte para apoiar a interconexão com computadores de grande porte. Porto Alegre, PGCC da UFRGS, 1986.

1v.

Diss. (mestr. ci. comp.) UFRGS-PGCC, Porto Alegre, BR-RS, 1986.

Dissertação: Redes de Computadores: Redes Locais: Protocolos: Padronização: Nível de Transportes: Máquinas de Grande Porte: Arquitetura.

A Jussara e Gabriela

AGRADECIMENTOS

Este trabalho tornou-se possível graças ao apoio e colaboração de um grande número de pessoas que, direta ou indiretamente, proporcionaram condições para a sua construção.

A todos externo meu agradecimento e, em especial, ao Centro de Processamento de Dados da Universidade Federal do Rio Grande do Sul, através de sua divisão de Computação, por ter me dado condições de iniciar esta empreitada, e às Lojas Renner, divisão de Processamento de Dados, permitindo sua continuação e fornecendo os equipamentos necessários.

Agradeço também à Profª Liane Tarouco pela orientação e apoio técnico, a Fernando Schlabitz e Juergen Rochol pela motivação recebida, a Maria Helena Cardozo e Lígia Castro pelo trabalho de datilografia e desenho, respectivamente.

Por fim, mas não menos importante, à minha esposa pelas horas de sono revisando os trabalhos e por ter permitido atrapalhar a sua própria dissertação.

SUMÁRIO

LISTA DE FIGURAS

RESUMO

Atualmente uma preocupação comum em todos os ambientes envolvidos em interconexões de computadores é criar mecanismos que possibilitem a interação entre programas contidos em diversos equipamentos. A ISO propôs uma forma padronizada para comunicações entre sistemas abertos e o nível de transporte é parte importante dessa padronização.

O objetivo deste trabalho é descrever a implementação de um serviço de transporte para uma máquina de grande porte com a intenção de utilizá-lo numa rede local de computadores.

Descreve-se no texto o modelo OSI e a proposta para redes locais, o padrão de transporte ISO classe 4 e os sistemas utilizados. Finalmente, é descrita a implementação num computador de grande porte.

ABSTRACT

Nowadays, a very common preocupation at several environments of computers interconnection is the creation of mechanisms which make possible the interaction between routines contained in different equipments. The purpose of ISO is a layered standard architecture for Open Systems Interconnection and the transport level is an important component of architecture.

The purpose of this work is to describe the implementation of a transport protocol to a large computer with a view to use in a local network.

It describes the OSI model and the local network standards, the ISO transport standard class 4 and the used system. At last, it describes the implementation for a large computer.

1 INTRODUÇÃO

Esta dissertação propõe-se a apresentar o estudo e conseqüente implementação de um protocolo de transporte padrão ISO classe 4, e as razões que justificam essa escolha, no intuito de conectar computadores de grande porte, modelo Burroughs, a redes locais.

Este trabalho iniciou-se no Centro de Processamento de Dados da UFRGS e continuou sua implementação nas instalações da Divisão de Processamento de Dados das Lojas Renner, servindo para a conexão entre computadores de grande porte.

O segundo capítulo descreve os protocolos utilizados em redes de computadores, o padrão OSI-ISO e sua definição, como também os padrões sugeridos para redes locais.

O terceiro capítulo corresponde ao nível de transporte. São apresentados alguns protocolos anteriores ou similares à padronização e, a seguir, o protocolo de transporte ISO: seus serviços, suas classes, a descrição das fases de estabelecimento de conexão, transferência de dados e encerramentos de conexão e seu diagrama de estados.

Já o quarto capítulo apresenta o ambiente da implementação, mostra as necessidades das máquinas de grande porte, os equipamentos utilizados na implementação, suas semelhanças e diferenças e alguns itens de software oferecidos por estes sistemas.

No capítulo cinco são definidas as conformações necessárias ao protocolo para uma implementação deste tipo e as primitivas funcionais que permitem a utilização do protocolo por um usuário qualquer ou pelo nível de sessão.

No sexto capítulo é descrita a implementação de fato do protocolo e as rotinas que compõem esta implementação, sua estrutura de dados, suas rotinas de atendimento ao

usuário e rotinas que perfazem o núcleo executor do diagrama de estados.

Por fim, o capítulo sete descreve os testes realizados, as funções de depuração. Também é mostrado um conjunto de medições que podem dar uma idéia do desempenho do sistema, e acaba sugerindo futuras implementações no sistema ora criado.

2 REDES LOCAIS E SEUS PROTOCOLOS

Não existe uma definição precisa para o significado do termo *redes locais* ou até mesmo de *redes de computadores*, todavia, podemos considerar como um conceito básico o de que uma rede de computadores consiste de um conjunto de computadores autônomos interconectados, isto é, dizemos que um ou mais computadores são ditos interconectados quando eles são capazes de trocar informação entre si. Não devemos exigir qualquer tipo de restrição quanto à forma física de interconexão, no entanto, os sistemas devem possuir autonomia entre si e, assim, excluiremos as conexões sistema central a terminais/micros escravos.

Uma rede de computadores heterogêneos pode ser definida como uma coleção de sistemas cujas características físicas e, sobretudo seus sistemas operacionais, diferem entre si. A *SNA* (System Network Architeture) admite, por exemplo, apenas conexões entre equipamentos IBM ou assemelhados, a *BNA* assim o faz com os BURROUGHS, a *DNA* com DIGITAL e assim por diante ... No caso dessas arquiteturas de rede, o usuário acaba por permanecer na dependência de apenas um fabricante devido à falta de compatibilidade entre os mesmos.

Após uma fase inicial, o uso de redes dirigiu-se para a interconexão de máquinas heterogêneas e veio a necessidade de uma padronização que assegurasse um mínimo de compatibilidade. A *ISO* (Organização Internacional de Padrões), então, criou um modelo de referência para sistemas cuja exigência era cooperação mútua entre processos pertencentes a equipamentos distintos. A esse tipo de sistemas chamou-se de *Sistemas Abertos* e, ao padrão, *Interconexão de Sistemas Abertos*, conhecido como modelo *OSI* /INT 80/.

2.1 O Modelo OSI

O modelo de referência OSI, objetivando definir o comportamento externo de um sistema aberto, apresenta uma visão lógica de sistemas interconectados, a arquitetura OSI, não implicando na estrutura de implementação do sistema. Mais tarde, com a aceitação deste modelo, foram especificados os padrões de protocolos OSI onde poderíamos,então, tornar possível a comunicação entre dois processos /INT 80/.

A estrutura básica da arquitetura consiste na sua subdivisão em camadas. De acordo com esta técnica, cada sistema é visto como um conjunto de subsistemas representados, por conveniência, em níveis verticalizados, conforme figura 2.1.

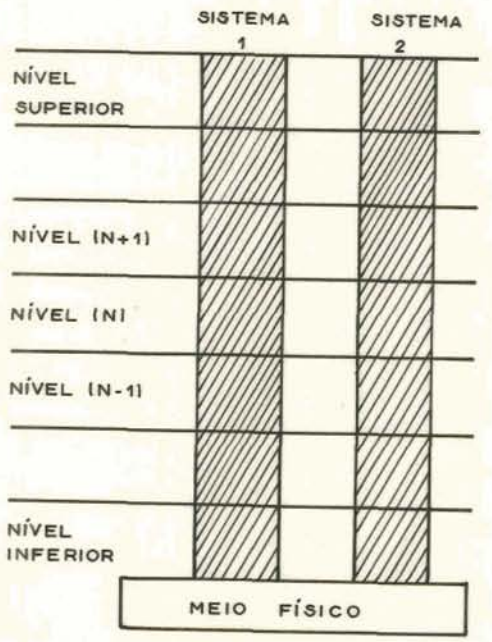


FIGURA 2.1 O modelo proposto

Tanenbaum /TAN 81/ cita as principais características que compõem a estrutura hierarquizada de sete níveis, ou camadas:

a) uma camada deve ser criada sempre que houver um nível de abstração diferente;

b) cada camada deve executar uma função bem definida;

c) a função de cada camada deve ser escolhida na direção dos padrões internacionais;

d) as camadas inferiores devem ser escolhidas de tal forma que minimizem o fluxo de informação entre os interfaces;

e) o número de camadas deve ser o suficiente para que funções distintas não sejam realizadas numa mesma camada mas não tão pequeno que a implementação da arquitetura se torne impraticável.

Podemos compreender, através da figura 2.1, que os níveis adjacentes comunicam-se através de interfaces comuns. Subsistemas e mesmo nível (n) formam a camada (n) da arquitetura. Um subsistema é composto de uma ou mais entidades, adjacentes na camada. As entidades da mesma camada são chamadas entidades par.

Simplificando, a uma entidade na n-ésima camada é chamada de entidade-(n). A entidade da camada superior será a entidade-(n+i) e a da camada inferior entidade-(n-1).

Cada entidade-(n) fornece à entidade-(n+1) um serviço dito serviço-(n). Devemos admitir que para cada camada são assumidos os serviços oferecidos pela própria camada assim como de suas camadas inferiores. Tais serviços são referidos entre camadas adjacentes via Pontos de Acesso ao Serviço.

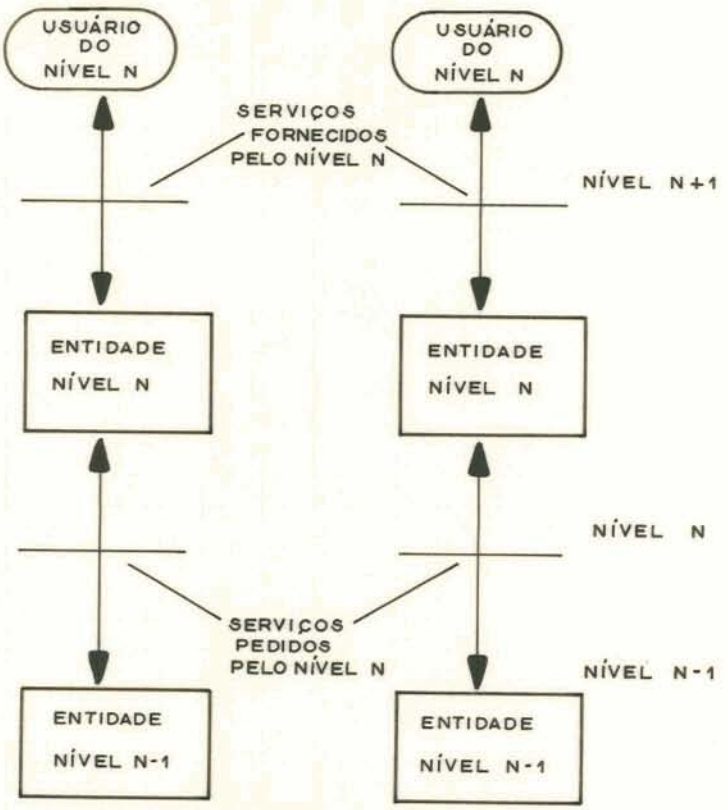


FIGURA 2.2 Visão estruturada do serviço de níveis

A operação de uma camada entre as entidades par é realizada pelos *protocolos* desta camada (para a camada-(n) um protocolo-(n)).

Em síntese, uma entidade-(n+1) requisita os serviços-(n) através do ponto-de-acesso-ao-serviço-(n) que permite à entidade-(n+1) interagir com a entidade-(n) na camada-(n). Daí, a informação será transferida, em vários tipos de unidades de dados, entre entidades par e entre entidades ligadas por um ponto-de-acesso-ao-serviço específico via um protocolo-(n): um conjunto de regras (semântica) e formatos (sintaxe) pelo qual informações e dados-(n) são trocados entre as entidades-(n) para a execução das funções-(n).

As funções-(n) são requisitadas e seus resultados retornados através de unidades-de-dados-do-serviço (n) (*UDSn*)

via seus pontos-de-acesso-ao-serviço e a sintaxe executada pelo protocolo-(n) e realizada através das unidades-de-dados-do-protocolo-(n) (*UDPn*).

A partir destas definições já é possível visualizar uma estrutura de níveis de um protocolo de transporte e seu relacionamento, demonstrado na figura 2.2.

2.1.1 Elementos que participam das fases do protocolo

2.1.1.1 Conexão

Uma conexão-(n) é o estabelecimento de uma associação para fins de comunicação entre duas ou mais entidades-(n+1) identificadas pelo seu endereço-(n), oferecido pela camada-(n) como um serviço, de maneira que a informação possa ser trocada entre as entidades.

Estabelecemos uma conexão quando oferecemos um endereço-(n) para a entidade-(n+1) fonte e um endereço-(n) para a entidade-(n+1) destino e associamos um com outro.

Desta forma, uma conexão nada mais é do que uma singularização das referências entre as entidades par, isto é, um estabelecimento de um endereço ou "assinatura" e a criação de um "enlace lógico" entre as duas.

Há três sub-funções deste serviço:

a) Estabelecimento de Conexão

O estabelecimento de entidades par de uma camada-(n) requer a disponibilidade intrínseca da conexão-(n-1) e, se isto ainda não ocorreu, esta deve ser primeiramente realizada e assim recursivamente iremos solicitando conexões inter-pares até o nível de uma conexão puramente física. Se não for possível realizar ao menos uma destas conexões procede-se da seguinte forma:

- as conexões superiores serão informadas do fato e poderão solicitar o encerramento das conexões inferiores;

- as conexões inferiores apenas apóiam e, portanto, não dependem das superiores.

Normalmente a fase de estabelecimento de conexão é composta dos seguintes procedimentos:

- requisição pela entidade-(n) do serviço de conexão à entidade-(n-1);

- requisição pela entidade-(n-1) à sua entidade par de uma conexão (estabelecimento de um elo via referências entre as duas);

- recusa ou aceite pela entidade par e sua informação à chamante;

- fornecimento à entidade-(n) do estado da conexão.

b) Liberação da Conexão

A liberação de uma conexão-(n) é o procedimento de desativar o elo entre as entidades par e suas referências anteriormente estabelecidas na fase de estabelecimento de conexão.

Considerando a sua definição, a existência da conexão-(n) desponta como uma premissa básica para a execução deste serviço. Outros requisitos são a conexão-(n-1) ou na ausência desta, e devido a esta ausência, uma referência temporal comum (*time-out*, etc.).

Um fato a observar é que a liberação da conexão-(n-1) não necessariamente encerra a conexão-(n) pois pode ser restabelecida ou substituída.

Após a entidade-(n) haver decidido liberar a co-

nexão ela pode tanto informá-la ao nível superior imediatamente como esperar um período que garanta a não mais existência de dados nesta conexão (a isto chamamos de tratamento de referências congeladas).

A fase de liberação da conexão deve ser composta, como regra geral, dos seguintes procedimentos:

- um evento que indique a necessidade de liberação da conexão-(n):

  . queda da conexão-(n-1),
  . encerramento por estouro de tempo,
  . solicitação do nível-(n+1);

- a informação à entidade par (se possível) da ocorrência e espera (se possível) da confirmação;

- o aviso ao nível superior da desconexão.

c) Multiplexação e Particionamento

A multiplexação da conexão-(n) é o controle e roteamento da conexão-(n) via uma ou mais conexões-(n-1). Isto pode ser realizado do seguinte modo:

- cada conexão-(n) é associada diretamente a uma conexão-(n-1), dito sem multiplexação;

- várias conexões-(n) via uma conexão-(n-1) no intuito de economizar serviços do nível inferior;

- uma conexão-(n) em várias conexões-(n-1), operação chamada *splitting* ou particionamento, no desejo de melhorar o rendimento na transmissão.

No último caso, é conveniente lembrar que os dados podem chegar fora de seqüência e há que utilizarmos algoritmos de resseqüenciamento e reordenação.

2.1.1.2 Transferência de dados

Os procedimentos relacionados com a transferência de dados são os procedimentos que visam a troca de informação real entre as entidades conectadas entre si.

Devemos observar que a palavra *informação*, neste contexto, possui um significado próprio. Informação, ou melhor, dados, consiste na unidade informação/dados na qual a camada-(n+1) solicita transferir através da camada-(n). Por sua vez, este conjunto de dados, dito informação, acrescido de dados de controle, constituirá a unidade de informação a ser passada à camada-(n-1).

Podemos verificar, então, que a transferência de dados consiste não apenas na transferência normal de dados como *todos* os procedimentos ou serviços que assegurem esta atividade bem como os demais, exclusivos de determinada camada, opcionais ou não.

a) Transferência normal de dados

As informações de controle e dados do usuário (no sentido explicado, anteriormente) são trocadas entre as entidades par através das *Unidades-de-Dados-do-Protocolo* - (n) (*UDP*n), isto é, uma unidade de dados especificada no protocolo-(n) contendo, presumivelmente, ambos tipos de dados transferíveis.

A informação de controle passada, informação-de--controle-do-protocolo-(n), deve ser transferida entre as entidades-(n) através de uma conexão-(n-1), assim como os dados que devem ser transferidos transparentemente.

Uma UDPn tem um tamanho fixo, finito, escolhido arbitrariamente e deve ser regida pelas normas do protocolo-(n).

b) Transferência de Dados Expressos

Dados expressos são dados onde o usuário deseja uma transferência ou um processamento prioritário aos serviços normais. Um exemplo é a utilização de mecanismos de interrupção por sinalização (*breaks*, etc.).

O fluxo dos dados expressos é independente do fluxo normal de dados. Podemos dizer que uma estação que suporte fluxo expresso possuiria dois canais, um para dados normais, outro para expressos cujo atendimento será prioritário.

c) Controle de Fluxo

Tais mecanismos, naturalmente, não devem estar atuantes em todos os níveis pois teríamos, então, uma redundância muito grande.

Falamos anteriormente que uma UDPn possui um tamanho fixo, no entanto, uma unidade-de-dados-de-serviço-(n) não, necessariamente, o terá. Logo, o mecanismo primário para o controle de fluxo será a segmentação destes dados, assunto tratado adiante. Sendo assim, aplicaremos o controle de fluxo já sobre unidades de dados fragmentadas, de um tamanho previamente conhecido.

Assim controlamos o fluxo no sentido par a par através de definições e procedimentos de protocolo, baseados no tamanho da unidade de dados ou entre camadas adjacentes (superior/inferior) baseados no tamanho da unidade de dados deste interface.

É interessante observar que a multiplexação numa camada inevitavelmente irá acarretar num mecanismo de controle de fluxo para cada conexão-(n-1) individualmente criada.

d) Segmentação e Concatenação

Conforme já demonstramos, os dados podem ser divididos em UDPns. A este procedimento, dá-se o nome de segmentação. Devemos, naturalmente, prover mecanismos que garantam a identidade destas unidades em relação à informação original. Ao procedimento que consiste na reaglutinação destas UDPn chamamos concatenação.

e) Seqüenciamento

Devido a diversos problemas que podem ocorrer no tráfego de dados nas camadas inferiores ou até mesmo naquela onde está o elo, nem sempre é possível garantir que os dados liberados cheguem na ordem em que isto ocorreu. Se à camada-(n), em atividade, for necessário que este ordenamento seja respeitado, devemos prover meios para que possamos identificar a seqüência correta dos dados transferidos preservando a original.

2.1.1.3 Tratamento de erros

Num meio onde transmitimos dados é natural esperar que ocorram erros, portanto, este padrão visa estabelecer os procedimentos para reduzi-los ou até eliminá-los.

a) Reconhecimento

Uma função de reconhecimento pode estar disponível no protocolo-(n) tal que não interfira no seguimento correto dos dados.

Este reconhecimento tanto deverá servir para verificar erros ocorridos na camada-(n-1) como na própria. O tratamento deve, por isto, tentar recuperar falhas outras como duplicação, segmentação ou seqüência errada. Podemos, inclusive, utilizar as mesmas informações de controle.

b) Detecção e Recuperação de Erros

Visam, além de prover maior confiabilidade ao meio, adequar sua performance não propagando, quando possível, aos níveis superiores.

É possível o uso de mecanismos de reconhecimento e até recuperação dependendo do nível de exigência do protocolo (do tipo janela, por exemplo).

2.1.2 Os níveis do protocolo

É difícil provar que uma estrutura de camadas em particular (como a utilizada pela IBM em sua *SNA*, DIGITAL na *DNA* ou a própria ISO) é a melhor solução possível, contudo, devemos nos ater a princípios que, quando aplicados, possam nos aproximar disto:

- não criar níveis em demasia, aumentando a complexidade do sistema;

- criar camadas separadas para manipulação de funções distintas ou de tecnologia diferente;

- colocar funções similares numa única camada;

- criar uma camada com funções facilmente localizáveis, de forma que esta se torne um único componente, facilitando alterações;

- criar uma camada onde já possuímos um nível diferente de abstração (sintaxe, semântica ...);

- criação de sub-camadas podendo, inclusive, serem "omitíveis".

Em sendo assim, a ISO propôs os seguintes níveis: físico, de enlace, rede, transporte, sessão, apresentação e aplicação.

O relacionamento destes níveis entre si é mostrado na figura 2.3.

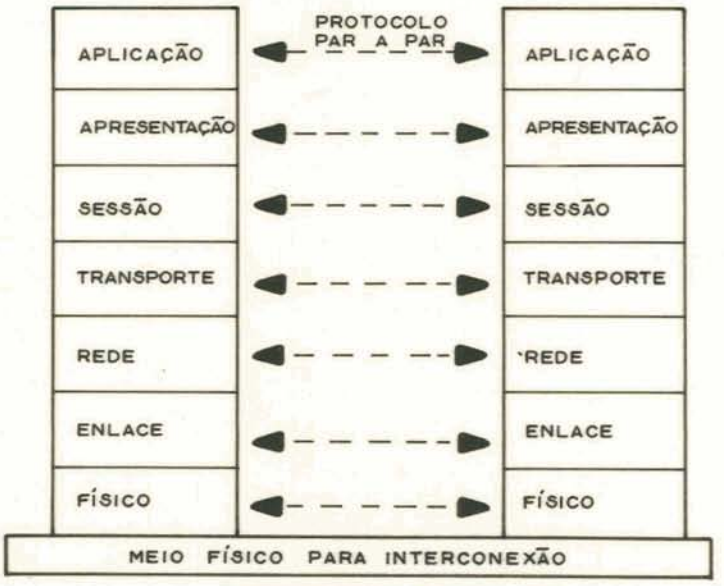


FIGURA 2.3 O modelo OSI

Veremos, agora, um esclarecimento sobre cada nível.

2.1.2.1 Aplicação

O nível mais alto, fornece serviços diretamente ao usuário, não a um nível superior. Seu propósito é servir uma janela entre os usuários comunicantes.

2.1.2.2 Apresentação

É a camada utilizada para representar a informação transitada numa forma *conveniada*, preservando seu significado. Isto inclui uma transformação de dados durante a viagem pelos níveis inferiores.

Tal transformação pode, por exemplo, ser usada para criptografia, compactação, etc., enquanto a formatação estabelece o conceito de *terminal virtual*.

2.1.2.3 Sessão

Fornece os serviços necessários para que as entidades de apresentação (a entidade-(n+1)) sincronizem-se e preparem seu diálogo, manipulando sua troca de dados.

Em algumas implementações o nível de sessão é omitido pois, se for implementado apenas o subconjunto de serviços correspondente ao Kernel ter-se-á uma mera replicação dos serviços do nível de transporte. Podemos dizer que o serviço básico da camada de sessão é a aposição de um serviço orientado ao usuário. Sincronização, *checkpoint*, recuperação também são tarefas passíveis deste nível.

Podemos, ainda, afirmar que a camada de sessão procura encapsular tarefas do serviço de transporte. Sempre que isto ocorrer, pode ser aberta nova requisição de transporte sem afetar a *sessão* do usuário.

2.1.2.4 Transporte

A função da camada de transporte é fornecer um serviço confiável independente da sub-rede (e suas idiossincrasias físicas) ao usuário. Possui primitivas para conexões, *telegramas* (dados expressos) e outros. É conveniente lembrar que existem dois tipos de protocolo de transporte:

- orientado a conexão;
- não orientado a conexão.

A camada de transporte, quando no uso do serviço orientado a conexão, identifica cada um de seus usuários pelo endereço de transporte (muitas vezes chamado *assinatura*), com este endereço é possível prover a interligação dos múltiplos processos de um sistema com múltiplos processos de ou-

tro sistema.

As funções deste serviço compõem-se de:

- mapeamento de endereço de transporte com o de rede;

- multiplexação de conexões de transporte através de uma conexão de rede;

- estabelecimento e término de conexões;

- controle de seqüência;

- detecção de erros e monitoração da qualidade do serviço;

- recuperação de erros;

- segmentação e concatenação;

- controle de fluxo;

- serviço de entrega de dados expresso (telegrama).

2.1.2.5 Rede

É a camada que realiza a conexão e transferência entre sistemas, operando a sub-rede de comunicações físicas.

Este nível garante ao transporte uma independência acerca do trajeto da mensagem e uma liberação desta, em teoria, sem erros. Assim permite que o nível de transporte seja voltado ao usuário final cabendo a si a tarefa de gerenciar o básico da comunicação.

2.1.2.6 Enlace

O nível de enlace fornece meios e procedimentos

para estabelecer, manter, liberar conexões de enlace entre entidades de rede, transformando-a numa linha isenta de erros de transmissões. Aqui, nós falamos de erros de linha, *checksums*, *delays*, etc.

2.1.2.7 Físico

Este nível fornece características funcionais elétricas e mecânicas para ativar, manter e desativar conexões para transmissão de bits entre as entidades de enlace. Aqui o que nos concerne são *volts*, tempo de bit, metros, etc.

2.2 A padronização de redes locais /BUR 84/

A rede local é um resultado da evolução do conceito de redes e de um rápido desenvolvimento de *hardware* tendo como conseqüência uma vasta gama de produtos. Esta situação, embora auspiciosa no tocante às possibilidades de utilização de recursos computacionais contudo indesejável quando da necessidade de conexão de produtos heterogêneos, gerou um grupo de estudo no Instituto de Elétricos e Eletrônicos resultando no *Projeto IEEE 802* /INS 83/ e /INS 82/.

Este projeto utiliza como referência o padrão ISO, mantendo os níveis implementados no OSI. Todavia, a norma não exige a implementação de todos estes níveis por parte do fabricante e, mais ainda, o projeto apenas padroniza detalhadamente os dois níveis inferiores.

Assim, o projeto IEEE 802 subdividiu o nível de enlace em dois subníveis, sendo que o inferior abrange também o nível físico.

FIGURA 2.4 Comparação dos modelos IEEE 802 e OSI-ISO

2.2.1 Subnível de controle de enlace lógico

O *LLC* (Logical Level Control) tem como função a troca de pacotes com o nível par. São previstas duas modalidades de serviço: o serviço não orientado a conexão, classe I e o serviço orientado a conexão, classe II /INS 82/.

Na classe I as unidades de dados são trocadas sem haver sido estabelecida uma conexão. As *UDPℓ* não necessitam ser confirmadas ("acknowledged") e não existe, ainda, controle de erros ou de fluxo.

Na classe II, podemos tanto operar na modalidade classe I como também estabelecendo conexão entre as entidades par. É realizado, ainda, um controle de erros e de tráfego.

2.2.2 Subnível de controle de acesso ao meio

Este subnível, na realidade, corresponde ao subní

vel inferior do nível de enlace bem como ao nível físico(veja figura 2.4). Sendo assim e levando em consideração o fato de existir várias finalidades e, sobretudo, vários meios distintos, o projeto IEEE 802 foi dividido em três padrões distintos conforme o meio físico, e a forma de acesso, explicados rapidamente a seguir.

2.2.2.1 CSMA/CD - IEEE 802.3

CSMA/CD - acesso múltiplo sentindo a portadora, com detecção de colisão. Este protocolo visa operar na forma difusão e é comumente utilizado em redes do tipo *Ethernet*. Sua forma de operação é a seguinte /INS 82/:

1. antes da transmissão, a estação verifica se o meio já não está sendo utilizado. Se estiver, aguarda um tempo aleatório, se não, transmite incondicionalmente;

2. quando a estação passar a transmitir, esta monitora o meio ouvindo e verificando se houve colisão (outra estação transmitiu simultaneamente). Se positivo, esta repete a operação após uma espera aleatória de tempo.

A verificação da colisão é muito simples de realizar. Basta "ouvir" o meio (um cabo coaxial) e verificar se a mensagem recebida é a mesma transmitida.

2.2.2.2 Token Ring - IEEE 802.5

Para redes com topologia em anel, as estações estão fisicamente encadeadas de tal forma que um pacote viaja pelo anel com um bit indicando se contém uma mensagem ou está ocupado. Se estiver disponível a estação pode, então, apor uma mensagem e ocupar tal bit.

Muito usado em redes industriais, pode trabalhar com fibras óticas obtendo alta performance. Todavia, é uma rede de alto custo, principalmente devido às soluções encon

tradas para aumentar a confiabilidade.

2.2.2.3 Token Bus - IEEE 802.4

O método de "passagem de bastão" (uma analogia com as corridas de revezamento) procura compatibilizar a confiabilidade do meio totalmente passivo de uma sub-rede independente que é o CSMA/CD com a eficiência e um conhecimento apriorístico do limite para um tempo de resposta que podemos obter numa rede em anel com passagem de bastão.

Neste protocolo as estações ao se incluírem na rede formam um anel lógico criando internamente uma tabela com estação anterior e estação posterior e a partir daí trabalham passando o bastão nesta seqüência lógica. Este protocolo prevê, ainda, a inclusão posterior de uma estação, sua saída da rede quer normalmente, quer por falha, refazendo as tabelas das estações vizinhas. Já foi chamado de "protocolo roda de chimarrão" e é visto como um "polling":

1. a estação detentora do bastão controla o meio por um período de tempo;

2. esta transmite até esgotar-se a fatia de tempo quando, então, entrega o bastão para a próxima segundo sua tabela.

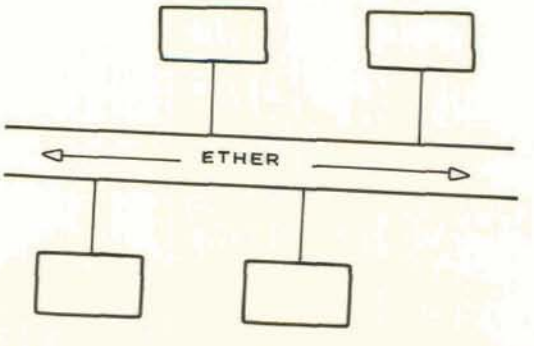


FIGURA 2.5 Rede apoiada pelo protocolo CSMA/CD

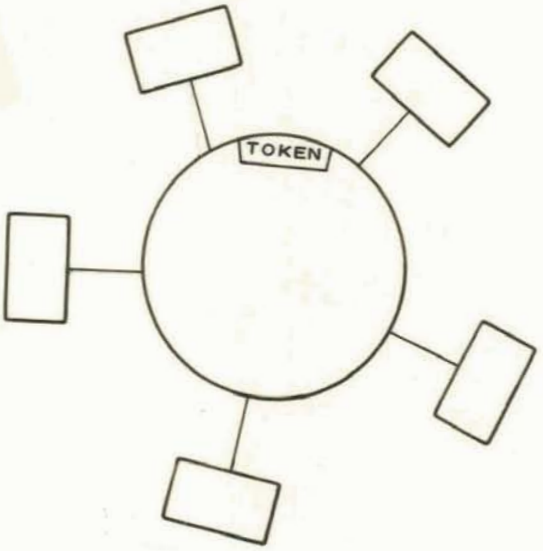


FIGURA 2.6 Rede apoiada pelo protocolo Token Ring

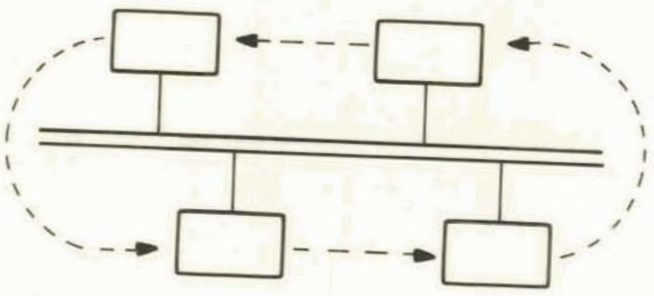


FIGURA 2.7 Rede apoiada pelo protocolo Token Bus
A seqüência lógica de passagem de bastão não necessariamente obedece o caminho físico.

3 O NÍVEL DE TRANSPORTE E SEU PROTOCOLO

Estamos esclarecendo a peça chave de uma arquitetura de comunicações entre computadores. Sempre existiu,portanto, uma demanda muito alta entre usuários, fabricantes e fornecedores de serviços de comunicação para um padrão mais efetivo do serviço de transporte pois é justamente este que vem a prover um mecanismo confiável para a troca de dados entre os diferentes processos de diferentes computadores assegurando a liberação dos dados sem erro, na seqüência correta, sem perdas ou duplicações, liberando o *software* de mais alto nível da tarefa de gerenciar o sub-sistema de comunicações /CRU 84/.

3.1 Alguns protocolos implementados ou propostos

- NCP/TCP da ARPANET

O original Network Control Protocol, baseado numa rede confiável acabou sendo substituído pelo Transmission Control Protocol, mais seguro e tolerável.

Ele aceita mensagens de até 65 Kbytes e envia cada uma num datagrama separado. O nível de rede não necessariamente garante uma liberação ordenada obrigando o TCP a manipular procedimentos de erro. Para tanto, o cabeçalho de cada mensagem é composto de 48 bytes com os mais diversos controles.

Esta enorme quantidade de campos com redundância fez-se necessária devido às características dos usuários da rede ARPA: pequenos computadores conectados aos mais diversos tipos de redes menores servindo à comunidade acadêmica, científica e exército.

- Camada de Transporte na SNA

Duas camadas realizam as funções básicas de trans

porte: *path control* e *transmission control*.

As conexões entre as entidades não são simétricas, tendo a primária maior prioridade que a secundária.

Também, nesta arquitetura, o protocolo de transporte ou o conjunto de camadas que o compõe baseiam-se numa sub-rede de confiabilidade baixa.

Por ser uma proposta IBM, este é um dos protocolos de transporte mais utilizados no mundo.

- Camada de Transporte da DECNET

É uma das mais simples implementações, sendo utilizada diretamente pelos programas do usuário. Seus comandos são dependentes da implementação e compõem-se de:

a) CONNECT REQUEST - uma requisição de conexão;
b) RECEIVE CONNECT - recepção de mensagens;
c) ACCEPT CONNECT - aceitar como válida uma proposta de conexão;
d) REJECT CONNECT - rejeitar a proposta de conesão;
e) SEND CONNECT - transmissão de mensagens.

- Padrão ECMA 72

Muito parecido com o padrão ISO, visto ser a Europa o conjunto de países mais influente nesta organização, este protocolo foi projetado de maneira modular usando o conceito de classes de serviços, acabou inspirando o padrão da ISO.

3.2 A opção pelo padrão ISO

Antes de mostrarmos a especificação escolhida ca-

be-nos tecer algumas considerações que ajudarão a dirigir ao protocolo em questão /MUS 85/.

- Protocolos de transporte tipo DECNET, SNA, BNA, só são compatíveis com as redes de seus próprios fabricantes, restringindo sua aplicação.

- O Departamento de Defesa americano, devido à complexidade e redundância, está abandonando o TCP como seu protocolo de transporte e dirigindo-se ao padrão ISO.

- O protocolo ECMA está praticamente abandonado pelos fabricantes europeus em prol do ISO.

- Num seminário/*workshop* patrocinado pelo Escritório Nacional de Padrões (NBS) do Departamento de Comércio americano os principais fabricantes e grandes usuários optaram, em redes locais, por usar um nível de rede nulo abaixo de um protocolo de transporte ISO classe 4; seminário este que redundou em demonstração conjunta no *NCC 84* /NAT 84/.

- A própria IBM, fabricante mais reticente, anunciou sua disposição em utilizar também os padrões ISO em paralelo a sua SNA.

- A DIGITAL, fornecedora da *Ethernet*, também está acompanhando os passos da IBM /BUR 84/.

- O Ministério das Comunicações baixou a portaria 772 de 14 de outubro de 1984 recomendando o uso de todos os padrões ISO /WEB 85/.

Além de todas estas afirmativas também nos resta a observação de ser o protocolo ISO de transporte aquele que atende ao maior número de redes e disponibilidade de recursos. Ora, numa máquina de grande porte (o objeto deste trabalho) onde há uma demanda por uma variada gama de serviços e conexões a máquinas do mais distinto porte, vem a calhar um protocolo que, como veremos a seguir, adapta-se aos mais diversos serviços e equipamentos.

3.3 O protocolo de transporte ISO

O serviço de transporte ISO fornece uma transferência transparente de dados entre seus usuários. Sua principal funcionalidade é permitir ao usuário que este ignore o meio utilizado para manipular seus dados /INT 84/ e /INT 83/.

São estes os principais serviços deste nível:

a) Seleção pela qualidade do serviço;

b) Independência dos recursos de comunicação das camadas inferiores;

c) Significação fim-a-fim;

d) Transparência da informação transmitida;

e) Endereçamento do usuário;

f) Transferência com prioridades de certas mensagens.

3.3.1 Primitivas do serviço de transporte

Conforme o padrão OSI, o usuário e o serviço comunicam-se através de primitivas funcionais, utilizadas nas três fases do serviço:

a) *Fase*: ESTABELECIMENTO DE CONEXÃO
Serviço: estabelecimento de conexão
Primitivas: T_CONNECTrequest (endereço chamante,
endereço chamado,
qualidade do serviço,
opção para dados expressos,
dados)
T_CONNECTindication(os mesmos anteriores)
T_CONNECTresponse(qualidade do serviço,
endereço do respondente,
opção de dados expressos,
dados)
T_CONNECTconfirm(os mesmos anteriores)

b) *Fase*: TRANSFERÊNCIA DE DADOS
Serviço: transferência normal de dados
Primitivas: T_DATArequest (dados do usuário)
T_DATAindication (dados do usuário)

Serviço: transferência expressa de dados
Primitivas: T_EXPEDITED_DATArequest (dados do usuário)
T_EXPEDITED_DATAindication (dados do usuário)

c) *Fase*: LIBERAÇÃO DE CONEXÃO
Serviço: encerramento de uma conexão
Primitivas: T_DISCONNECTrequest (dados do usuário)
T_DISCONNECTindication (razão da desconexão,
dados do usuário)

3.3.2 Relacionamento das primitivas e os usuários

É natural que uma primitiva acabe causando uma reação no usuário da outra entidade par. As figuras abaixo ilustram estes procedimentos, já com suas abreviaturas usuais. Note que incluímos alguma temporização.

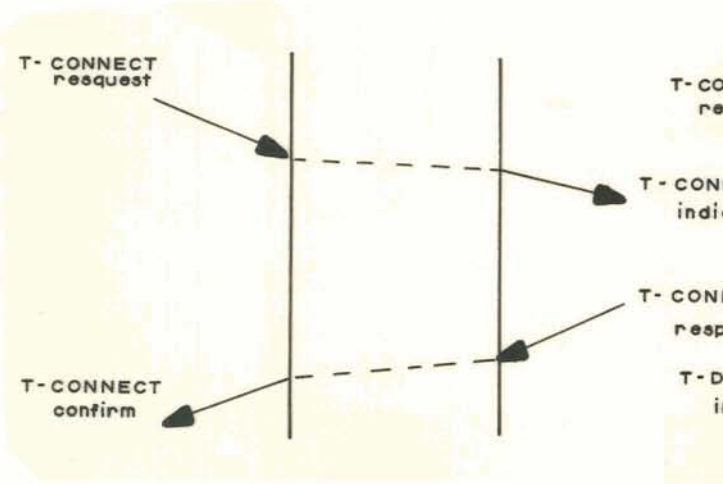


FIGURA 3.1 Estabelecimento de conexão bem sucedido

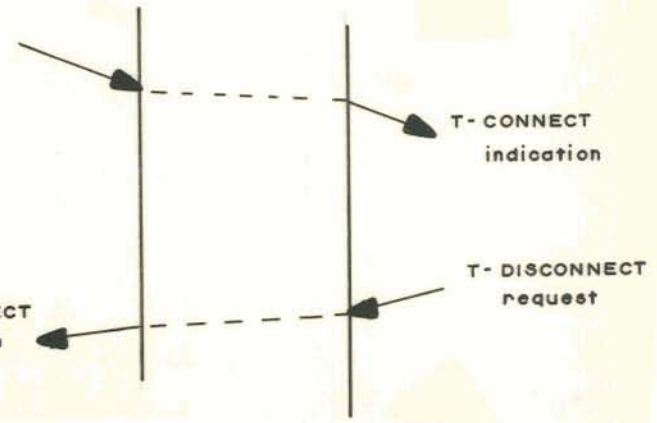


FIGURA 3.2 Rejeição, pelo usuário, de uma conexão

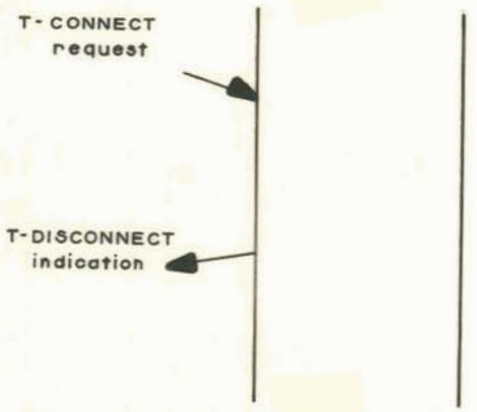


FIGURA 3.3 Rejeição pelo serviço de transporte de uma conexão

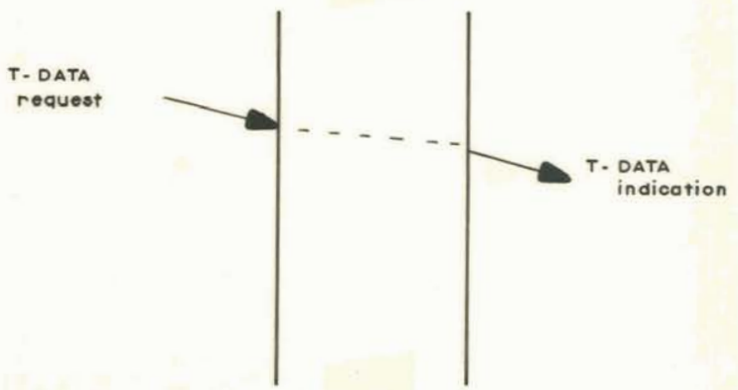


FIGURA 3.4 Transferência normal de dados

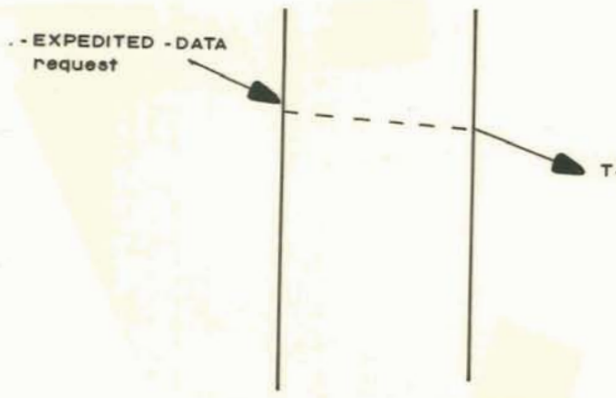


FIGURA 3.5 Transferência expressa de dados

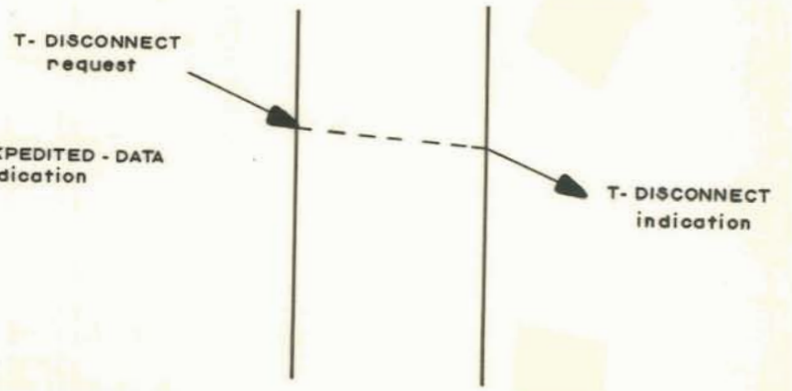


FIGURA 3.6 Desconexão realizada pelo usuário

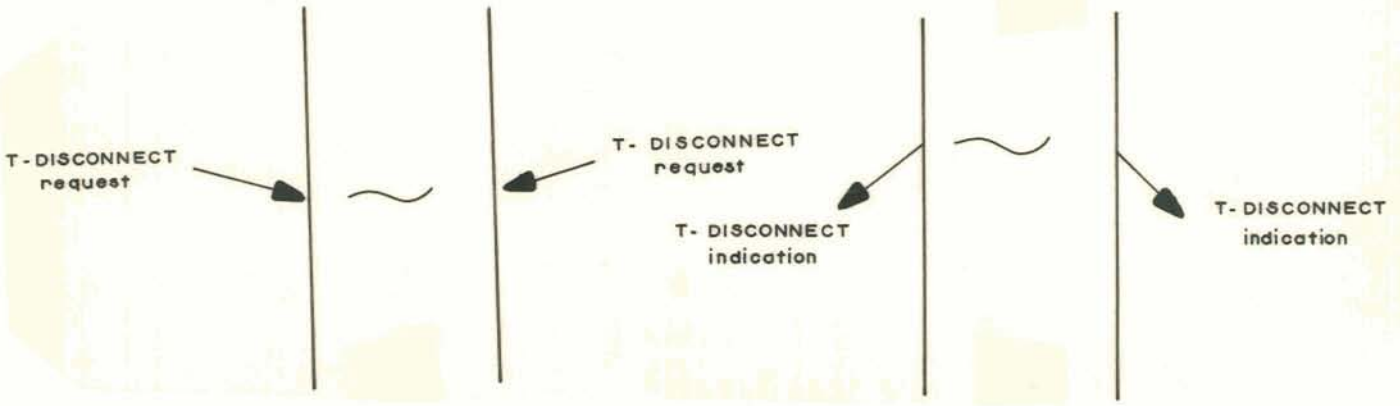


FIGURA 3.7 Desconexão realizada por ambos usuários

FIGURA 3.8 Desconexão realizada pelo serviço de transporte

3.3.3 Diagrama global do serviço de transporte

A partir disto, podemos descrever um macrodiagrama de estados para seqüências possíveis do serviço de transporte.

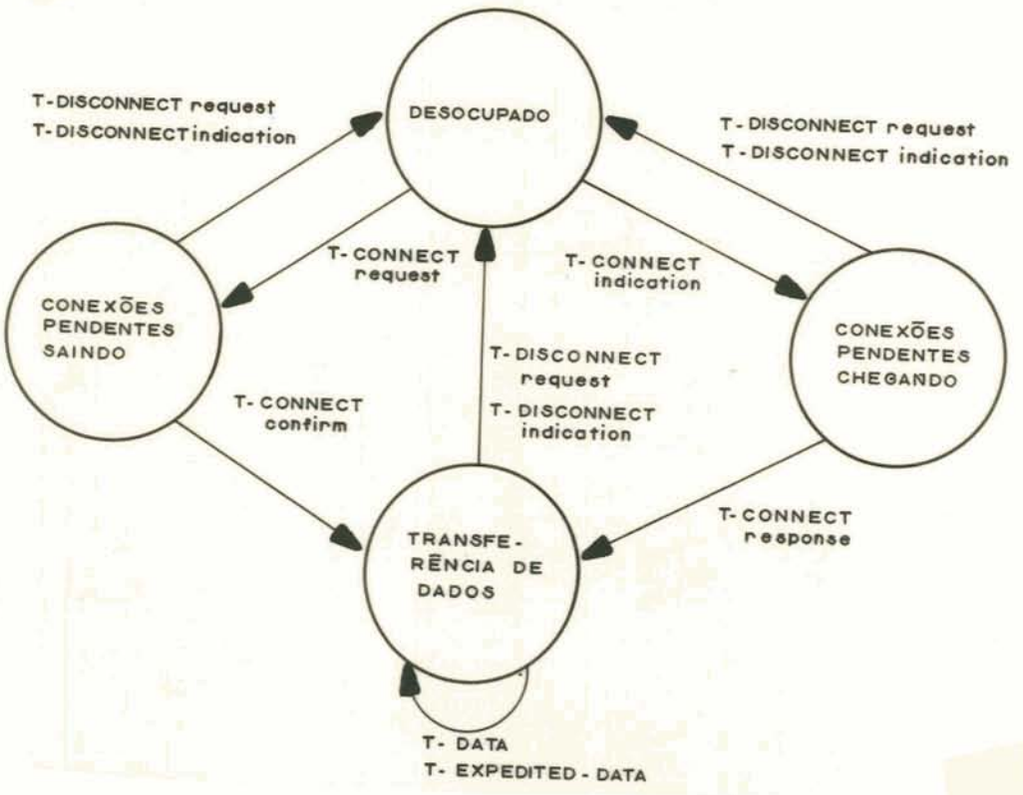


FIGURA 3.9 Diagrama de estados global do serviço de transporte

3.4 Classes do protocolo ISO de transporte

Para atender a tantos usuários distintos, a camada de transporte foi dividida em classes:

a) classe Ø: classe simples;

b) classe 1: classe com recuperação básica de erros;

c) classe 2: classe com multiplexação;

d) classe 3: classe com multiplexação e recuperação de erros;

e) classe 4: classe com detecção e recuperação de erros.

Estas classes intentam trabalhar sobre tipos distintos de sub-redes. Naturalmente, a complexidade da classe será diretamente proporcional à má qualidade da sub-rede. A ISO definiu três tipos:

a) tipo A: taxa de erros residuais aceitável e taxa de erros sinalizados também aceitável;

b) tipo B: taxa de erros residuais aceitável mas alta taxa de erros sinalizados;

c) tipo C: taxas inaceitáveis em ambos os tipos de erro.

3.4.1 Características da classe Ø

É a classe mais simples, utilizada para apoiar a recomendação S.70 do CCITT para *TELETEX*. Foi projetada para operar apenas em redes do tipo A.

3.4.2 Características da classe 1

Seu principal propósito é recuperar-se de um *reset* ou desconexão de rede. Portanto, sua seleção é realizada com base em critérios de confiabilidade.

Projetada para operação em redes tipo B.

3.4.3 Características da classe 2

A classe 2 multiplexa várias conexões de transporte sobre uma única conexão de rede tipo A.

Na fase de abertura desta classe é possível negociar o uso ou não de controle de fluxo explícito. O uso deste controle visa evitar o congestionamento entre as conexões de transporte. Isto é aconselhado, por exemplo, quando o tráfego é pesado e contínuo como em sistemas bastante multiplexados. O uso de controle de fluxo pode otimizar tempos

e recursos, todavia, na situação oposta e no intuito de diminuir a sobrecarga, podemos não desejar tal facilidade.

3.4.4 Características da classe 3

A classe 3 consiste dos serviços da classe 2 mais a capacidade de recuperar-se após um *reset* ou desconexão de rede. Está para a classe 2 assim como a classe 1 está para a classe Ø. Com isto, podem operar redes do tipo B.

3.4.5 Características da classe 4

Fornece as características da classe 3 mais detecção dos erros que possam ocorrer devido ao baixo grau de qualidade da rede (tipo C, por exemplo).

Os tipos de erros detectados incluem: perda de pacotes, liberação fora de seqüência, duplicação ou corrupção de dados.

Devido a sua grande gama de utilização e as possibilidades de comunicar-se com implementações de seus subconjuntos, ela torna-se a classe ideal para implementações em equipamentos de grande porte, definindo a nossa escolha.

Como já efetuamos nossa decisão, doravante, discutir-se-á apenas a classe 4 usando, contudo, conceitos relativos ao protocolo como um todo.

3.5 Primitivas do serviço de rede apoiando o transporte

Existem cinco primitivas que realizam o apoio à condução do serviço de transporte /INT 83/:

a) N_CONNECT solicita, indica, confirma uma nova conexão de rede;

b) N_DATA trata de enviar e receber dados;

c) N_DATA_AK reconhecimento de dados;

d) N_EXPEDITED_DATA tráfego de dados expressos;

e) N_RESET solicitação ou aviso de um *reset* de rede;

f) N_DISCONNECT solicitação ou indicação de encerramento de conexão.

3.6 Especificação da classe 4

Como já vimos, a classe 4 fornece a funcionalidade da classe mais a habilidade de detectar e recuperar-se de perdas, duplicações ou erros de seqüência de UDPTs sem a necessidade de uma participação do usuário (o nível superior). Esta detecção de erros faz-se por meio de mecanismos como *time out*, numeração e outros procedimentos. Além disto, utiliza *checksum* para controle de transmissões com erro.

Veremos adiante os principais procedimentos desta classe e uma indicação de seu uso como obrigatório ou não na implementação /INT 84/.

3.6.1 Temporizadores

O padrão não define valores de temporização específicos e as descrições deste item não são obrigatórias:

a) M - Vida de uma UDSR - um limite máximo para a duração entre a transmissão de uma UDSR pelo transporte e a recepção na entidade par (UDSR, unidade de dados do serviço de rede);

b) E - Retardo máximo de trânsito esperado - retardo máximo limite sofrido por todos exceto uma pequena porção de UDSRs;

c) $A\ell$ - Tempo de reconhecimento local - tempo máximo que leva entre a recepção de uma UDPT e a transmissão de seu reconhecimento correspondente pela entidade local;

d) Ar - Tempo de reconhecimento remoto - idêntico a Al só que para a entidade remota;

e) Ti - Tempo de retransmissão local - máximo tempo em que a entidade de transporte esperará pelo reconhecimento;

f) R - Tempo de persistência - tempo máximo que uma entidade local continuará transmitindo a UDPT;

g) N - Número máximo de retransmissões;

h) L - Limite sobre referências e números de seqüência - tempo máximo entre a transmissão de uma UDPT e a recepção de qualquer reconhecimento relacionado a ela;

i) I - Tempo de inatividade - limite de tempo após o qual a entidade de transporte desconecta-se se não recebeu nenhuma UDPT;

j) W - Janela de tempo - tempo máximo para uma entidade decidir reenviar a última UDPT no sentido de evitar que a conexão seja liberada por tempo de inatividade (item i).

Conforme for considerado na implementação,há duas maneiras de controlar os temporizadores de retransmissão:

1ª - um intervalo é associado a cada UDPT, se o tempo expirar, ela será retransmitida;

2ª - um intervalo de tempo é associado à conexão de transporte, se ocorrer estouro de tempo, toda a janela será retransmitida. Se houver o reconhecimento de uma UDPT de dados que não a última o temporizador será reiniciado a partir deste ponto.

3.6.2 Estabelecimento de conexão

Se não houver, o serviço de transporte deve ativar uma conexão de rede e, após, executar os procedimentos para o estabelecimento ou rejeição de conexão.

Lembre-se que uma conexão não é considerada estabelecida até que haja uma troca completa de três UDPTs, isto é, CR, CC e AK.

A entidade iniciante de transporte deve enviar uma UDPT do tipo CR contendo um conjunto de parâmetros que irão definir a conexão em si. A entidade chamada responde com uma UDPT CC, caso aceite a conexão, informando parâmetros com valores diferentes se não puder, ou desejar, aceitar os propostos pela iniciante. Por fim, a iniciante envia, ainda, uma UDPT AK confirmando o estabelecimento da conexão.

Os seguintes campos devem ser trocados:

a) referências - cada entidade de transporte escolhe uma referência (um "endereço de transporte" ou "assinatura" de 16 bits diferente de zero e com valor arbitrário);

b) endereços (opcional) - indica os pontos de acesso ao serviço, se para cada endereço de rede existir um único serviço de transporte então este não será necessário;

c) crédito inicial - o tamanho da janela;

d) tamanho da UDPT;

e) versão do protocolo;

f) parâmetro para segurança - um código qualquer utilizado pelo nível superior para controle de segurança em *login* ou acesso;

g) tempo de reconhecimento;

h) taxa de erro residual aceitável;

i) prioridade;

j) retardo de trânsito;

l) tempo para reassinalamento.

A entidade chamada pode recusar-se a estabelecer a conexão utilizando para tanto a UDPT DR, onde num campo apropriado, conterá a razão.

3.6.2.1 Assinalamento a uma conexão de rede

O iniciador deve assinalar uma conexão de transporte e, no mínimo, uma conexão de rede já existente ou solicitar uma nova.

Uma conexão de rede sem conexões de transporte pode existir apenas no intervalo entre o pedido da entidade iniciante e o aceite pela entidade solicitada.

É de bom tom que apenas o proprietário de uma conexão de rede possa liberá-la. Neste caso, a entidade par da rede deve comunicar à entidade de transporte a que serve e, esta, se não conseguir refazer a conexão, deve liberar todas as conexões de transporte associadas àquela conexão de rede. Se houver particionamento o provedor do serviço de transporte deve verificar se existem mais e encerrar todas as conexões de rede atendendo apenas àquela conexão de transporte.

3.6.3 Transferência de dados

A transferência de dados faz uso do temporizador de inatividade para precaver-se de paradas por falhas na conexão ou na própria entidade par, mantendo um intervalo de tempo W para retransmissões no caso de sua atividade, garantindo ao par que ela ainda existe.

3.6.3.1 Dados normais

A entidade de transporte utiliza UDPTs DT para a transmissão de dados exigindo uma confirmação da entidade receptora. Este controle é feito através de campos contendo a ordem destas UDPTs (seqüência: TPDU_NR) que serão reconhecidas por esta seqüência no UDPT AK, de confirmação.

Adiante, veremos outros procedimentos que envolvem mais detalhes desta fase.

3.6.3.2 Dados expressos

A entidade de transporte utiliza dados expressos através da UDPT ED, que deverá ser numerada (campo TPDU_NR) distinta das UDPTs de dados normais (DT).

A entidade receptora transmitirá uma UDPT EA com a mesma seqüência no campo YR_EDTPDU_NR. Se isto não ocorrer, a entidade transmissora irá repetir a UDPT ED, não podendo jamais enviar uma UDPT de seqüência seguinte se a em questão não for reconhecida.

Este tratamento é feito desta forma para assegurar que UDPTs ED sejam liberadas pelo usuário em seqüência e que o usuário receptor não as receba mais que uma por vez. Também garante a chegada desta UDPT antes da de dados (DT).

3.6.3.3 Seqüenciamento

A entidade receptora deve liberar ao usuário UDPTs DT conforme a ordem indicada nos seus campos de seqüência, não liberando aqueles que virem fora de seqüência até chegar seus predecessores. Os procedimentos neste caso são livres de normas.

Com isto, podemos detectar UDPTs duplicadas não reliberando-as mas reconhecendo-as pois esta duplicação pode redundar numa perda de um AK. AK este que conterá a mes-

ma seqüência da UDPT que ele estiver confirmando. Um AK fora de seqüência deve ser totalmente ignorado.

3.6.3.4 Controle de fluxo

Após um crédito inicial enviado (janela) a entidade pode controlar o fluxo através destas janelas quando as UDPTs são confirmadas ou, ainda, gerando o campo CDT da UDPT AK.

A qualquer tempo será possível enviar um novo AK modificando o campo acima e, conseqüentemente, liberando a entidade par para novas transmissões.

3.6.3.5 *Checksum*

Usado para detectar corrupção de UDPTs na rede, utiliza dois octetos, calculados pela seguinte fórmula:

$$\sum_{i=1}^{L} a_i \equiv \emptyset \text{ (módulo 255)}$$

$$\sum_{i=1}^{L} ia_i \equiv \emptyset \text{ (módulo 255)}$$

onde i = número do octeto na UDPT;
a_i = valor do octeto na posição;
L = tamanho da UDPT em octetos.

3.6.3.6 Segmentação e remontagem

Consiste em dividir uma unidade inteira de dados do serviço de transporte em várias UDPTs para compatibilização ou economia. A entidade de transporte deve, então, ma

pear estes dados em UDPTs sobre uma seqüência que não deve ser prejudicada por outra UDPT.

O parâmetro EOT (campo da UDPT DT) indica se ocorreu fim ou não deste conjunto de dados.

3.6.3.7 Concatenação e Separação

Permite a conexão de transporte usar conexões múltiplas provendo proteção adicional contra falhas de redes ou para aumentar o desempenho da comunicação.

Temos que observar que a solução para a liberação e chegada corretas de UDPTs é resolvida através do mecanismo de seqüenciamento.

3.6.4 Liberação de conexão

A liberação normal é feita através das UDPTs DR (Disconnect Request) e DC (Disconnect Confirm). Quando ocorrer a transmissão de uma destas, a entidade deve realizar, também, a desconexão de rede (veja parágrafo 3.6.2.1).

3.6.5 Associação de UDPTs a Conexões de Transporte

É o procedimento realizado ao interpretar uma UDSR como contendo UDPTs e, se possível, associar cada uma destas UDPTs com uma conexão de transporte.

3.6.5.1 Identificação

Se uma UDSR não puder ser decodificada, então a entidade de transporte deve ignorá-la apondo um *reset* à rede ou desconectá-la. Se a decodificação for possível mas a UDSR estiver corrompida, esta deve ser ignorada pois haverá um tratamento no nível de rede quer por meio de temporizadores, quer por meio de tratamento comum de erros.

No caso positivo, após a separação de cada UDPT a

entidade de transporte irá tentar a associação com alguma conexão de transporte.

3.6.5.2 Associação de UDPTs individuais

Se a UDPT for um CR (Call Request) e for duplicada esta deverá ser associada com a conexão criada pelo valor original da UDPT CR, se não for duplicada, será criada uma nova conexão de transporte.

Em todas as demais UDPTs, o campo DST_REF indicará qual a conexão de transporte a identificar.

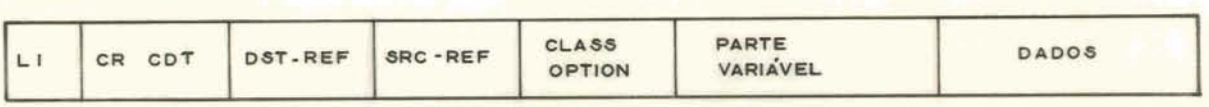
Se o campo DST_REF não referenciar nenhuma conexão de transporte, a entidade de transporte deve enviar uma UDPT DR se aquela tiver sido uma UDPT CC com DC se for DR e deve ignorá-la para os demais tipos.

3.7 Tipos e campos de uma UDPT

Cada UDPT possui um campo de código indicando sua natureza. Pode estender-se de meio a um octeto.

Uma UDPT é composta de uma parte fixa contendo seu código e tamanho do cabeçalho (LI), de uma parte variável composta de no mínimo três octetos (o primeiro, código do parâmetro; segundo, tamanho e terceiro em diante, o próprio parâmetro) e, opcionalmente, dados.

3.7.1 Requisição de conexão (CR)



Utilizada para informar à entidade par o desejo de abrir uma conexão de transporte. Seus vários parâmetros são propostos para *modelagem* da conexão.

a) código: 111Ø;

b) CDT: alocação inicial de crédito (abertura de janela);

c) DST_REF: zero;

d) SRC_REF: a referência da conexão indicada pela entidade de transporte originadora;

e) CLASS e OPTION: classe deve ser 4 e a opção indica o desejo da entidade requerente de usar ou não controle de fluxo explícito.

Como parte variável podem ser utilizados diversos parâmetros, opcionais:

a) TSAP_ID: identificador do ponto de acesso ao serviço;

b) TPDU-SIZE: tamanho desejado da UDPT;

c) VERSION: a versão implantada para o protocolo;

d) SECURITY: parâmetros usados, para controle de segurança;

e) CHECKSUM: controle de paridade (16 bits);

f) classe do protocolo alternativa para a comunicação;

g) tempo para reconhecimento;

h) desempenho desejado;

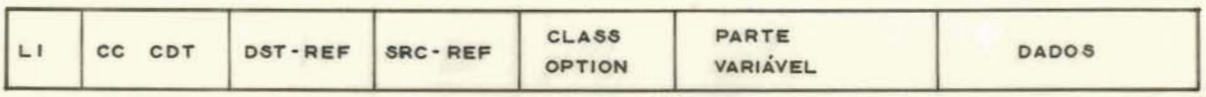
i) taxa de erro residual;

j) prioridade;

l) retardo de trânsito;

m) tempo de reassinalamento.

3.7.2 Confirmação de conexão (CC)



Indica que a entidade par aceitou a solicitação de conexão. Os vários parâmetros informam, ou o reconhecimento proposto por um CR, ou uma nova intenção de negociação.

Contém, basicamente, os mesmos campos do CR, no entanto, DST_REF deve ser igual à SRC_REF fornecida pelo CR e sua SRC_REF será a referência criada pela entidade de transporte que gerou o CC.

Seu código é 1101.

3.7.3 Solicitação de desconexão (DR)

Utilizada no tráfego comum para encerrar uma conexão ou quando a entidade de transporte rejeita uma solicitação de conexão (CR).

É composta, basicamente, de:

a) código: 1000 0000;

b) REASON: razão para desconexão que pode ser:

- desconexão normal iniciada pela entidade de sessão ou outro nível superior;

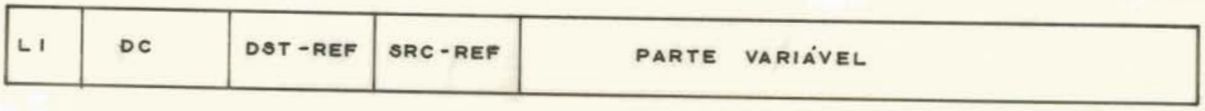
- congestionamento da entidade remota durante estabelecimento de conexão;

- negociação acabou discordante;

- erro de protocolo;

- outros.

3.7.4 Confirmação de desconexão (DC)



É uma resposta a uma solicitação de desconexão.

Seu código é 1100 0000.

3.7.5 Dados (DT)

É a UDPT utilizada para a transferência de dados

entre entidades de transporte. Seus campos:

a) código: 1111 0000;

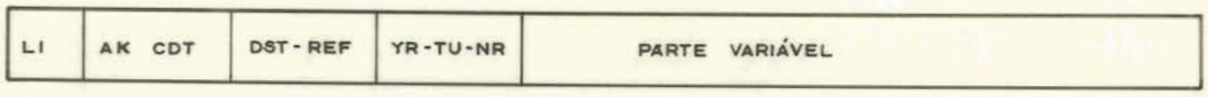
b) EOT: um *bit* que, quando valendo 1, indica que esta UDPT é a última unidade de dados de uma seqüência de DTs (que formam uma UDST);

c) TPDU_NR: o número de seqüência deste tipo de UDPT.

O tamanho do campo para dados é negociado no estabelecimento da conexão, daí a necessidade de uma segmentação dos dados.

Ainda, durante a transferência de dados, pode-se, mediante a parte variável, reduzir o tamanho deste campo.

3.7.6 Reconhecimento de dados (AK)



Esta UDPT é utilizada pela entidade de transporte para indicar a seu par que recebeu dados corretamente.Além disto, supre três outras funções básicas:

a) informar um recebimento de confirmação de conexão;

b) transmitir, quando expirado o tempo de janela, para avisar à entidade par que ainda há atividade;

c) alteração de crédito.

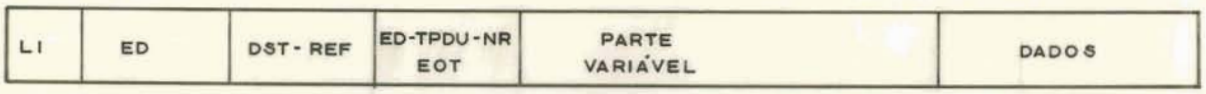
Seus campos:

a) código: 0110 xxx

Os quatro *bits* à direita (CDT) são utilizados para informar o crédito que a entidade par ainda tem para transmitir;

b) YR_TU_NR: número de seqüência indicando qual seqüência deverá ter a próxima UDPT de dados.

3.7.7 Dados expressos (ED)



Fornecida quando se é desejado enviar "telegramas" ou dados urgentes. Deve ser atendida prioritariamente, não podendo, jamais, haver mais de uma UDPT ED pendente.

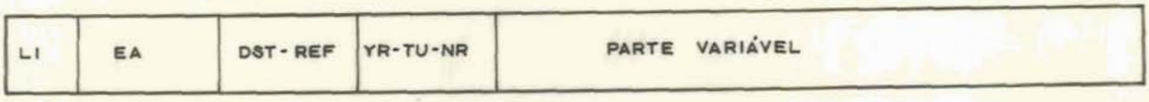
Seus campos:

a) código: 0001 0000;

b) ED_TPDU_NR: número de seqüência da UDPT ED;

c) EOT: fim da UDST, como na de dados normais.

3.7.8 Reconhecimento de dados expressos (EA)



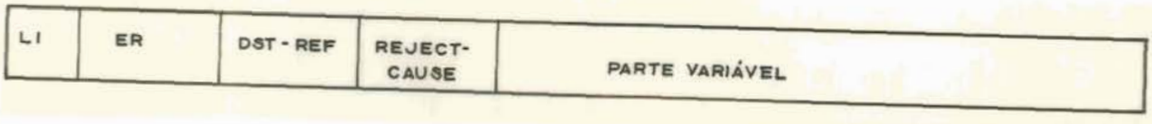
Atua conforme um AK no reconhecimento de dados expressos.

Seus campos:

a) código: 0010 0000;

b) YR_EDTU_NR: número de seqüência da próxima UDPT ED.

3.7.9 Erro (ER)



Usada quando ocorrer um erro de protocolo ou outro não previsto.

Seus campos:

a) código: 0111 0000;

b) REJECT_CAUSE: a causa do erro, que pode ser:

- não especificado;
- código de parâmetro inválido;
- tipo de parâmetro inválido;
- valor do parâmetro inválido.

3.8 Estados do protocolo

Uma maneira mais esclarecedora de descrever um protocolo é demonstrá-lo através de um diagrama de estados, com eventos saindo e chegando.

Antes de definirmos exatamente cada estado podemos, então, conceituar cada evento.

3.8.1 Eventos que chegam à entidade de transporte

Há três categorias de eventos que, chegando, podem causar alguma alteração nos estados da conexão:

a) Eventos causados pelo usuário, isto é, as primitivas de requisição utilizadas pelo mesmo quando deseja solicitar ou informar algo ao transporte; compõem-se de:

- TCONreq (requisição de uma conexão);
- TCONresp(resposta à requisição de uma conexão);
- TDTreq (requisição de transmissão de dados);
- TDISreq(requisição de desconexão).

b) Eventos causados pelo serviço de rede, informando a troca de alguns estados da conexão deste nível; podem ser:

- NDISind (indicação de desconexão de rede);
- NCONconf(confirmação à solicitação de conexão de rede);
- NRSTind (indicação de um *reset* na rede).

c) As UDPTs provindas da rede.

3.8.2 Eventos causados pelo serviço de transporte

São aqueles eventos que afetam o usuário ou à rede:

a) Eventos dirigidos ao usuário:

- TCONind (indicação de uma nova conexão);
- TCONconf(confirmação da abertura de uma conexão previamente solicitada);
- TDTind (chegada de dados);
- TEXind (chegada de dados expressos);
- TDISind(indicação de fim de desconexão).

b) Eventos dirigidos ao nível de rede:

- NDISreq (solicitação de fim de conexão de rede);
- NRSTresp(resposta a um *reset* de rede);
- NCONreq (solicitação de nova conexão de rede).

c) As UDPTs transmitidas pelo serviço de transporte.

3.8.3 Eventos causados pela temporização

São os eventos ocorridos quando a expiração de tempo em algum estado:

a) T_RETRANS (temporizador para retransmissões);

b) T_REF (temporizador para as referências congeladas);

c) T_INACT (temporizador para registro de inatividade);

d) T_WINDOW (temporizador de janela inativa - proteção para informar que a inatividade não é devida a fa-

lhas e sim por uma verdadeira inatividade temporária).

3.8.4 Estados

São estes os estados do protocolo de transporte ISO:

3.8.4.1 Fase de estabelecimento de conexão

a) CLOSED - ainda não há conexão de transporte;

b) WFNC - esperando uma conexão de rede;

c) WBCL - esperando um evento antes de liberação definitiva; no caso, espera uma UDPT CC para, após, enviar uma UDPT DR;

d) WFCC - esperando uma UDPT CC (confirmação de conexão);

e) WFTRESP - esperando que o usuário aceite a conexão proposta, através da primitiva T_CONNECT response;

f) WFCC-R - esperando por uma UDPT CC, havendo um reassinalamento em progresso.

3.8.4.2 Fase de transferência de dados

a) OPEN - conexão de transporte aberta, atividade normal, recebendo e transmitindo dados;

b) WFEA - esperando reconhecimento por um EA;

c) SCLOSEDW - janela de transmissão fechada (crédito esgotado);

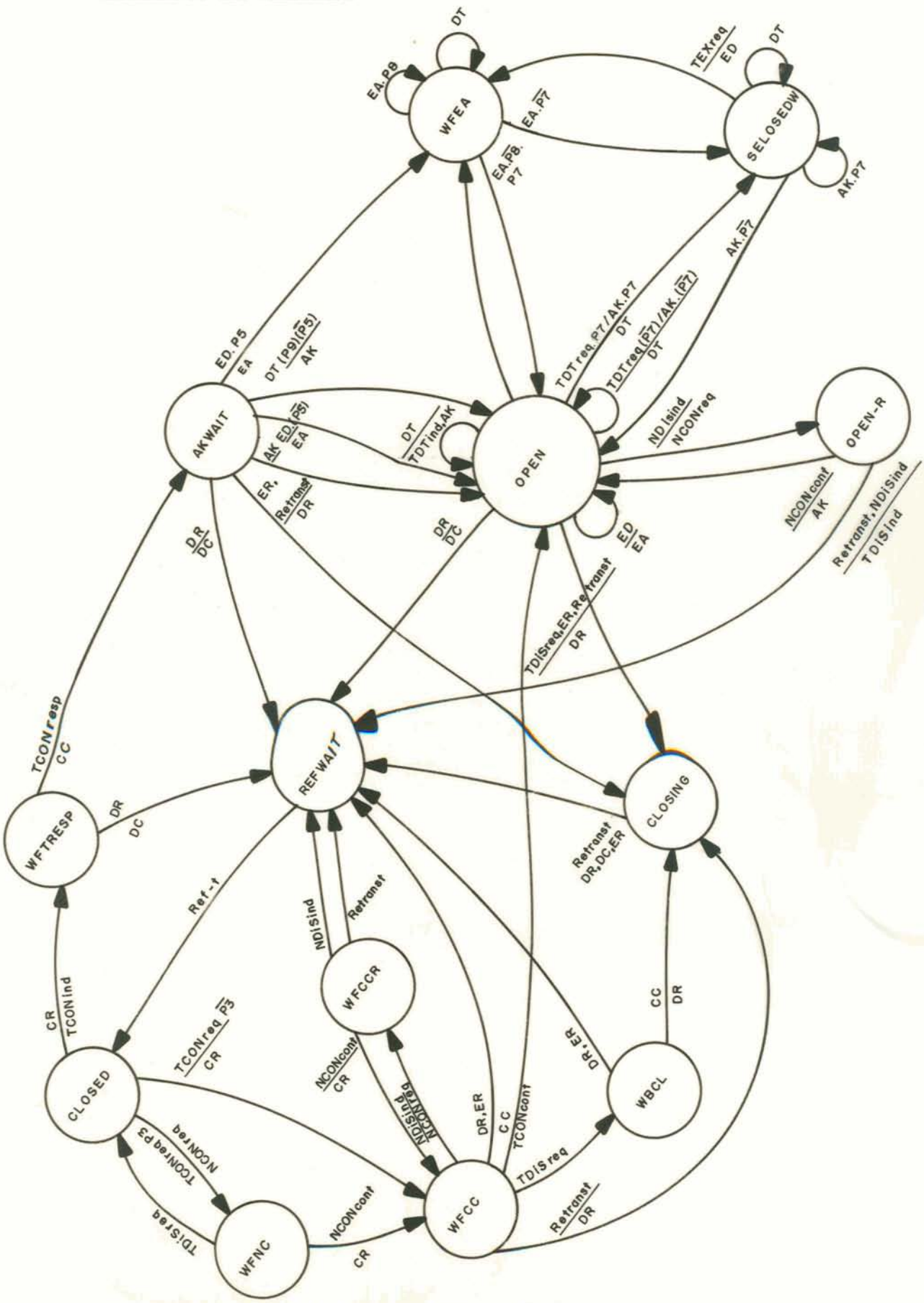
d) OPEN_R - conexão aberta, embora aguardando reassinalamento.

3.8.4.3 Fase de encerramento de conexão

a) CLOSING - liberação em progresso;

b) REFWAIT - esperando um período, T_REF, para suplantar referências congeladas.

3.9 Diagrama de estados



4 O AMBIENTE DA IMPLEMENTAÇÃO

Até agora foi discutido o protocolo com o qual desejamos trabalhar; todavia, nada foi falado acerca dos equipamentos que darão suporte a esta tarefa. Trabalhando em paralelo com outra dissertação, pretendemos operar numa rede local que contenha não só microcomputadores mas um espectro muito maior de equipamentos. Dentre eles, destacamos máquinas de porte médio ou superior (grupos 4 e 5 segundo a classificação SEI), pois é o objeto do nosso trabalho.

Infelizmente, não existe um padrão para sistemas operacionais para esse tipo de equipamento, como seria o CPM para microcomputadores de 8 bits, devido à sua própria natureza complexa e porque existem singularidades que acabam por representar um mérito ao fabricante. Destarte, somos obrigados a deter-nos num único tipo de equipamento deixando implementações similares em outros equipamentos.

Dois motivos nos levaram a optar pelas máquinas Burroughs:

a) DIGITAL e IBM já estão preparando sua própria implementação do modelo ISO podendo tornar inútil parte do nosso esforço;

b) o principal motivo, a disponibilidade do equipamento, tanto na UFRGS com o B6700 (já fora de linha), como na RENNER com um B6910 e um A9-F (seu modelo mais recente), totalmente compatíveis entre si.

4.1 Máquinas de grande porte

Não podemos desejar que uma máquina de grande porte possua as mesmas características de um microcomputador, possuindo, cada um, características próprias comentadas a seguir /DEI 84/.

Tais máquinas, com um mínimo de alguns megabytes

de memória não possuem grandes limitações de recursos, por outro lado, há uma grande concorrência entre seus diversos processos. Esses recursos devem então ser compartilhados e de tal forma que estejam garantidos todos os mecanismos de exclusão mútua. Para tanto, o sistema operacional deve ser um sistema que execute alguma forma de gerenciamento de memória ("swaping", paginação, etc.), um controle de vários processos concorrentes, rotinas de "spooling" e gerenciamento de discos de alta velocidade, além de periféricos especiais. Ao programa de aplicação não deve ser permitido acesso direto a esses recursos pois tal redundaria numa perda de controle do ambiente pelo sistema operacional e, por isto, apenas solicita a este sistema, sob forma de primitivas ou chamadas ao supervisor, operações reais de entrada e saída, tornando-a o mais transparente possível.

Toda máquina de algum porte possui uma rede de teleprocessamento. Naturalmente pode atender a diversos protocolos com um grau aceitável de transparência ao usuário. Tais rotinas são anexas ao sistema operacional e este encarrega-se também do controle efetivo desta comunicação. Daqui não podemos excluir a comunicação máquina a máquina em velocidades bastante altas.

Os programas de aplicação variam em sua gama de atividade, tanto podem ser listadores comuns, calculadores ou processos on-line acessando um banco de dados. Essa implementação visa tornar possível a comunicação entre processos não residentes em uma mesma máquina.

Normalmente, ainda são disponíveis rotinas e ou processos padrões que qualquer programa ou usuário pode solicitar, tais como cópia de fitas, manipulação de arquivos, edição e formatação de programas, visões de bancos de dados, etc.

Como isto pode ocasionar um risco muito alto em sistemas multiusuário como o em questão, tornam-se necessá-

rios mecanismos de segurança de tal forma que os usuários não afetam os demais e, ainda, controle do acesso a não autorizados. Isto normalmente é feito através de contas e senhas. Outros mecanismos adicionais podem ser: controle de erros de programas, arquivos especiais com múltiplas senhas, esperas de segurança, etc.

Em vista de ser permitido vários usuários e seus processos estarem sendo executados simultaneamente, não é possível utilizar "block time" para contabilização dos mesmos. Por isso carece-se de uma contabilização automática, em tempo real dos recursos utilizados em cada processo.

Devido às necessidades de desempenho é interessante estudar a arquitetura dos equipamentos nos quais será implementado o protocolo.

4.2 Os equipamentos utilizados na implementação

Esta implementação foi realizada em três equipamentos da linha Burroughs: o Burroughs B6700 da UFRGS, já fora de linha, e na RENNER um B6910 (isto é, o B5200 americano) e o A9, o mais recente lançamento.

São três configurações totalmente distintas, tanto a que se destinou como mesmo em parte:

a) B6700:

- 0,7 Mips
- 3 Mbytes de memória
- 1,1 Gbytes de disco
- 4 fitas
- 2 impressoras
- 32 linhas de comunicação
- ano de fabricação: 1972
- objetivo: uso geral, time-sharing, produção batch e compilações.

b) B6910:

- 0,65 Mips
- 6 Mbytes de memória
- 0,6 Gbytes de disco
- 3 unidades de fita compartilháveis com o A9
- 1 impressora
- 8 linhas de comunicação
- ano de fabricação: 1982
- objetivo: desenvolvimento de software.

c) A9-F:

- 4 Mips
- 12 Mbytes de memória
- 2,6 Gbytes de disco
- 2 unidades de fita compartilháveis com o B6910
- 1 impressora
- 12 linhas de comunicação
- ano de fabricação: 1985
- objetivo: produção, 22 sistemas on line, grandes bancos de dados.

Portanto, é conveniente o estudo desta linha de equipamentos, o que é facilitado, devido às semelhanças de arquitetura e sistema operacional.

4.3 A linha de máquinas de grande porte Burroughs

Esta linha de equipamentos notabilizou-se já há cerca de vinte anos como uma máquina não Von Neumann utilizada em operações comerciais.

Dijkstra e outros defenderam a idéia de construir um computador baseado em requisitos de software, orientado a uma linguagem específica e após então é que seria projetado o hardware. A Burroughs, encampando a idéia, construiu o B5500, um produto considerado revolucionário, baseado na

linguagem Algol e, para tal, uma máquina de pilha.

Este equipamento não só trazia o conceito de pilha como novidade mas sim uma série de características muito modernas, nas quais algumas ainda não foram implementadas em outros computadores. Podemos citar /BUR 84a/:

a) O conceito de pilha por processo

A cada processo existe uma área de trabalho denominada Stack onde os registradores são manipuladas na forma de pilha, com registradores de hardware orientados nesse sentido.

b) Separação entre código e dados

A grande diferença, o código-objeto estava localizado em área que não a dos dados do programa, o que facilitou mecanismos de controle e proteção.

c) Reentrância

Devido à separação entre código e dados é possível a várias cópias do programa utilizar a mesma área de código, reduzindo memória e processamento de controle.

d) Segmentação

No sentido de gerenciar memória virtual, utiliza-se a divisão das áreas de memória em segmentos ao invés de partições (na época) ou paginação. Este segmento corresponde a blocos de Algol.

e) Sistema de interrupções

Toda entrada e saída é feita através de interrupções, melhor que através de seleção e pesquisa, desafogando a UCP e utilizando-se de outro processador para este tipo de operação.

f) Spooling automático.

g) Busca automática de volumes.

O sistema operacional escrito para controlar tamanha gama de "novidades" foi escrito numa linguagem especial, o ESPOL (neste equipamento não existem montadores, apenas compiladores), parecido com o Algol, só que especializado para a escrita deste tipo de software. A este sistema chamou-se MCP-Master Control Program, mais tarde reescrito em NEWP, que lembra Modula.

Uma característica interessante do MCP é que este pode executar em dois estados:

a) normal, interrompível, executa rotinas solicitadas pelo usuário como a porção lógica de entrada e saída, controle de eventos, etc.;

b) controle, onde não pode ser interrompido e é utilizado um determinado bloco para controle de processos e operações "físicas" de entrada e saída, etc.

Com isto criou-se um sistema operacional com um Kernel não interrompível e agregado que com o tempo foi crescendo de funções, algo parecido com o Unix e, com isto, atualizado até hoje.

Como arquitetura podemos dizer que ela possui suas funções modularmente distribuídas onde ao MCP pertencem funções como o subsistema de entrada e saída, formalização, arquivos porta, controle da comunicação de dados, e outros módulos principais. É interessante destacar os procedimentos afetos à nossa área.

4.3.1 O Subsistema de comunicação de dados

Fisicamente é composto de processador central mais um front-end chamado Network Support Processor (NSP) cuja

função é realizar o tratamento desde o nível físico até o de enlace. Basicamente ele controla as disciplinas de linha /BUR 84a/.

Sendo um processador à parte o NSP é programável através da linguagem NDL II (no B6700 a linguagem chama-se NDL e o front-end DCP) - Network Definition Language II especialmente projetada para este fim. NDL II é composta de módulos /BUR 82/:

- Algorithmin: é a lógica do protocolo de enlace de dados;

- Editor: equivalente à camada de apresentação, faz a edição do texto;

- Configuration: define características físicas das conexões e equipamentos.

Após a mensagem ser tratada no NSP através do programa em NDL II é, então, entregue à memória principal e tratada pelo DCC.

O DCC (Data Comunnication Controller) é o elemento do MCP que apropria os dados trafegados no NSP. As mensagens vindas do NSP são colocadas numa área de memória chamada RESULT QUEUE. Já as mensagens que o sistema central intenta transmitir são colocadas na REQUEST QUEUE.

Outra função do DCC é atender solicitações das MCSs (veremos adiante) através das chamadas DCWRITE, um conjunto de funções especiais de controle de rede.

Quando o DCC examina a REQUEST QUEUE ele apanha todas as mensagens sem interrupção. Cada mensagem pode conter dados ou apenas controle; se for do primeiro caso, o DCC insere uma mensagem na fila do MCS que gerencia a estação para posterior direcionamento.

O MCS (Message Control System) é um programa do sistema que controla um conjunto de estações. Embora possa haver mais de um por sistema, a cada estação, só é permitido atribuir um MCS por sistema. É um programa escrito em DCALGOL e tem por responsabilidade a segurança e controle das estações, controle de arquivos remotos, manipulação de erros e alocação de recursos. É o interface com os programas de aplicação. Podemos citar como MCSs,o GEMCOS,CANDE e o X25MCS.

Abstraindo em relação ao padrão OSI, a arquitetura Burroughs é muito anterior, pode definir o programa escrito pelo NDL como nível de enlace e o MCS como nível de rede.

4.3.2 Arquivos porta

Existem diversas formas de realizar a comunicação entre processos; todavia, se for executada através do uso de arquivos do tipo PORT (ou Porta) será possível fazê-la sem variáveis comuns e todo o envolvimento necessário que o uso concorrente exige.

Podemos comparar estes arquivos com o "pipe" do UNIX em que dois ou mais processos que desejam comunicar-se entre si criam uma ponte ou um duto, entre eles, onde fluem as mensagens de interesse /CUN 85/.

No caso, quando um processo "escreve" na porta este está enviando a mensagem ao seu par que a receberá quando "ler" a porta. Um processo, ou processos, pode esperar por uma mensagem através do comando READ, que causará a suspensão do processo até que outro programa realize a gravação para este particular programa.

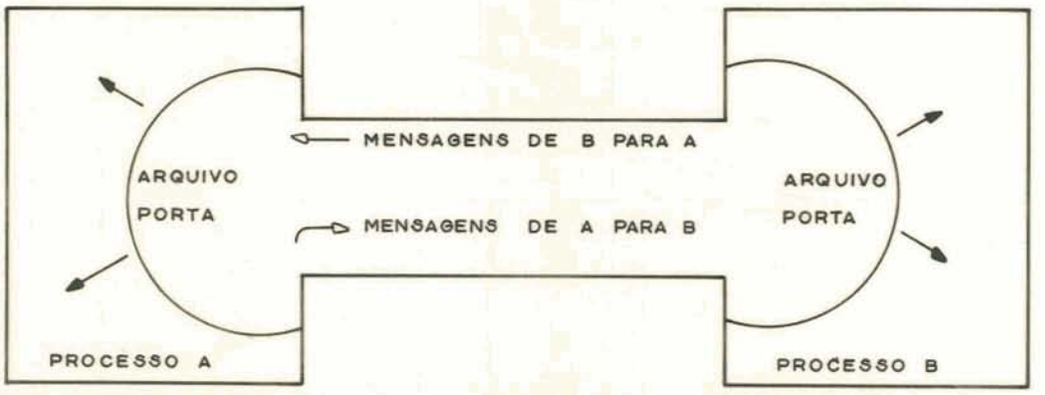


FIGURA 4.1 Visão lógica do interface de comunicação

Os arquivos do tipo PORTA diferem dos outros tipos de arquivos em que o equipamento físico está associado com o arquivo lógico. Em um arquivo PORTA todas as operações de entrada e saída ocorrem entre arquivos lógicos declarados em diferentes programas, ou no próprio, através de um ou mais subarquivos (subfiles). Assim, uma porta é constituída de um ou mais subarquivos, cada um destes podendo ser conectado a um diferente processo, através dos atributos de arquivo: TITLE (título), MYNAME (nome do processo), YOURNAME (nome no outro processo) e outros opcionais/BUR 84b/.

4.3.2.1 Abertura de um arquivo porta

Um subarquivo provém de um caminho lógico de comunicação entre dois processos na forma ponto a ponto e em duas mãos. Para estabelecer este canal, cada programa deve descrever a ligação desejada por meio do casamento dos atributos YOURNAME do subarquivo com o MYNAME da porta no outro

processo. Além, o TITLE de ambos deve também ser o mesmo.Esta operação é chamada de casamento (matching).

O que acontece é que quando o comando OPEN é executado para um subarquivo, o estado deste é mudado para "oferecido" e o algoritmo de casamento procura um subarquivo complementar, quando, aí, abre realmente o arquivo trocando seu estado para aberto.

Quando da abertura, é requerido um parâmetro, o subarquivo a ser aberto (que sempre será indicado por um índice, caso contrário, referir-se-á a todo o arquivo) e um segundo, chamado opção de abertura:

a) ESPERAR - o subarquivo é oferecido para casamento e aguarda, suspenso, tal ocorrência;

b) OFERECER - o subarquivo é oferecido para casamento e o programa não espera que o arquivo complementar seja encontrado;

c) TORNAR DISPONÍVEL - o subarquivo é oferecido e procura-se um subarquivo já oferecido, todavia se não encontrar, não haverá erro.

4.3.2.2 Fechamento de arquivo

Os subarquivos são fechados explicitamente ou quando no encerramento do programa que o abriu.

A operação de fechamento de um arquivo pode ser realizada de forma assíncrona.

4.3.2.3 Leitura e gravação

Portas e seus subarquivos podem ser lidos e escritos síncrona ou assincronamente. A bem disto há atributos de arquivos que facilitam operações de entrada e saída, como, por exemplo:

a) CENSUS - número de mensagens na fila, aguardando atendimento;

b) INPUT EVENT - evento que pode ser esperado, sendo causado quando chegou mais alguma mensagem na fila;

c) CHANGE EVENT - evento que informa troca de estado de um arquivo;

d) etc.

Exemplo de um programa em COBOL /BUR 84c/

```
ID DIVISION.
ENVIRONMENT DIVISION.
I-O SECTTION.
FILE CONTROL.
   SELECT P ASSIGN TO PORT.
DATA DIVISION.
FILE SECTION.
   FD   P.
    Ø1  REG   PIC X(80).
PROCEDURE DIVISION.
S SECTION.
P.
CHANGE ATTRIBUTE MYNAME OF P TO "MEU NOME".
CHANGE ATTRIBUTE YOURNAME OF P TO "SEU NOME".
OPEN P.
P1.  IF  P.CENSUS GREATER THAN Ø
         READ P
            AT END DISPLAY "FIM"
                   GO FIM
            ELSE DISPLAY REG
     ELSE WAIT ATTRIBUTE INPUTEVENT OF P.
               ATTRIBUTE CHANGEEVENT OF P.
            GO P1.
FIM. STOP RUN.
```

4.3.3 Biblioteca de execução

É comum nas instalações em geral existir diversas rotinas de uso comum a vários de seus sistemas. Uma solução encontrada para minimizar esforços duplicados é a criação de rotinas de biblioteca que podem ser de duas formas:

a) copiadas e anexadas ao fonte do programa;
b) ligadas ao objeto gerado.

No caso do sistema MCP tais bibliotecas acompanham o segundo tipo. Todavia a diferença dos demais sistemas operacionais,elas são ligadas em tempo de execução,aproveitando-se da reentrância de seus objetos.

Especificamente, uma biblioteca ("Library") é um programa comum que fornece um conjunto de pontos de entrada ("Entry Points") que podem ser chamados por outros programas /BUR 85/.

Qualquer usuário pode criar e utilizar uma ou mais bibliotecas. O compartilhamento destas é possível não só a diversos programas como a diversos usuários. Além disto é possível o uso de globais.

O programa de Biblioteca fornece seus pontos de entrada que são distintos das subrotinas comuns através do comando EXPORT e se torna efetivamente uma biblioteca após o comando FREEZE e é permitido, então, procedimentos de inicialização.

Um programa do usuário (o chamante) usa estes pontos de entrada e seus parâmetros. Para tanto ele deve apenas indicar o nome da biblioteca e as subrotinas desejadas. Uma biblioteca pode chamar outra mas não a si mesma.

Exemplo de uma Biblioteca:

```
BIBLIOTECA DE NOME BIB, EM ALGOL

BEGIN
  INTEGER PROCEDURE FATORIAL (N);      % ROTINA EXPORTADA
    INTEGER N;
    IF N<1 THEN FATORIAL := 1
    ELSE FATORIAL := N * FATORIAL (N-1);
  LIBRARY OUTRABIBLIOTECA (TITLE = "UM TÍTULO");
  STRING PROCEDURE DIADASEMANA (DATA) % CHAMADA INDIRETAMENTE
    INTEGER DATA;
    LIBRARY OUTRABIBLIOTECA;

  EXPORT FATORIAL,
         DIADASEMANA;  % MESMO QUE ESTEJA EM OUTRA
  DISPLAY "BIBLIOTECA BIB SENDO DISPARADA";
  FREEZE (PERMANENT); % JÁ EXECUTOU O "DISPLAY"
END.


PROGRAMA EM COBOL, USUÁRIO

ID DIVISION
    .
    .
    .
CALL "DIADASEMANA OF BIB"  USING  UMA-DATA
                           GIVING O-DIA.
    .
    .
    .
CALL "FATORIAL OF BIB" USING VALOR
                       GIVING FATORIAL.
    .
    .
    .
```

Veja que ao contrário do ALGOL, em COBOL não é necessário a pré-declaração da biblioteca e tampouco do ponto de entrada.

5 A IMPLEMENTAÇÃO NUMA MÁQUINA DE GRANDE PORTE

Se levarmos em conta a multiplicidade de recursos que um equipamento de grande porte tem a oferecer, fatalmente desejaremos conectá-lo não só a uma rede mas, sobretudo, a diversas e não necessariamente homogêneas. A mescla envolverá desde redes públicas de baixa confiabilidade e performance de transmissão até aquelas locais com dados completamente íntegros.

Para que tal ambientação se torne possível carece uma estrutura de rede na qual é possível a abordagem por duas soluções: o projeto de vários protocolos de transporte ou apenas um universal /MUS 84/.

Na primeira proposta é procurado um ajuste fino às características de cada rede, otimizando tanto quanto possível no trato com elas. Esta opção traz consigo a desvantagem de uma multiplicação um tanto quanto exagerada de programas que constituam os provedores do nível de transporte, além de, necessariamente, exigir do usuário ou do implementador do nível de sessão um conhecimento dos vários serviços oferecidos ou uma sobrecarga ao sistema na tentativa de compatibilizar os diferentes pontos de entrada e saída.

A segunda alternativa exige a implementação de uma classe de transporte completa o suficiente como a quatro (a nossa eleita). Devemos considerar que, quase que obrigatoriamente, já haverá uma classe deste porte pois é pouco provável a não conexão deste equipamento a uma rede do tipo C, trazendo consigo a vantagem da unicidade tanto em relação ao usuário bem como no relacionamento entre os vários transportes de cada máquina.

O nível de transporte é um processo que exige pesada temporização tornando mais cômodo "embutir" o protocolo no software básico para um tratamento adequado. Além des

ta provisão, tal feito permite a execução de seus serviços através de primitivas funcionais residentes no sistema operacional utilizando de mecanismos tipo "pipe" do UNIX ou Porta para a comunicação entre usuários, processos e protocolos.

Se observarmos detalhadamente a definição do serviço de transporte haveremos de notar que há um período no qual a conexão habita uma espécie de limbo. Isto é, ela já existe mas ainda nada ocorreu com ela. É necessário, então, para sua criação, um evento ou primitiva que possibilite isto, do tipo "Estou aqui e quero utilizar uma conexão de transporte ainda não sei para quê!". Adiante trataremos disto /TAU 85/.

No padrão ISO não estão previstas duas funções,contabilização e segurança, fundamentais para sistemas de grande porte. Uma solução a ser anotada é implementar parte do protocolo de transporte sobre controle do sistema operacional e parte, além dos níveis superiores, em rotinas de bibliotecas contabilizadas no processo do usuário, desde que a transferência seja mantida /LAN 84/.

Devido ao fato de que a maior parte das conexões a uma máquina de grande porte intentam usá-la como servidor apenas é útil que o protocolo seja capaz de reconhecer endereços públicos de recursos especializados e tratá-los de uma maneira singular /BRA 84/.

5.1 Primitivas funcionais /AND 84/

Convém lembrar que não necessariamente será obrigatório o uso de tais primitivas, apenas como ponto de acesso aos serviços do protocolo isto torna-se facilitado.

Para sua utilização o usuário deve utilizar um arquivo do tipo Porta que será o interface de comunicação entre ele e os demais integrantes da rede /PAL 84/.

5.1.1 Apresentação-ao-transporte

PARÂMETRO: um Arquivo Porta

RETORNO: -1 se houve algum problema
Endereço da conexão

Esta primitiva deve ser utilizada pelo usuário (que pode ser o nível de sessão) para "apresentar-se" ao serviço de transporte. Seu parâmetro é um arquivo Porta,que deve estar fechado, onde, após procurar por uma conexão livre esta é associada ao arquivo ficando numa espécie de limbo. Sempre deverá ser a primeira primitiva a ser chamada.

5.1.2 Conecte

PARÂMETROS: Arquivo Porta
Endereço: Inteiro
Processo par: Texto
Espera Retorno: Lógico
Código de Segurança: String(opcional)

RETORNO: <0 se não foi possível conectar, senão o índice da conexão

Esta primitiva pode ser utilizada no intuito do usuário do serviço de transporte comunicar ao provedor do serviço seu desejo de conectar-se a um processo par residente num sistema endereçado pelo parâmetro endereço. Caso o processo par esteja vazio então, na realidade, o usuário ficará aguardando até que alguém conecte-se a ele, sendo retornado o endereço e, em processo par, o nome do processo chamante como se fôra realizado por este.

Espera retorno é uma variável que se verdadeira indica o desejo de parar o processo até que haja uma comunicação e conseqüente conexão completada. Se isto não ocorrer durante um determinado período CONECTE retornará um valor negativo. Haverá forma de, assincronamente, descobrir se a

conexão foi realizada.

Código de segurança é um parâmetro opcional e serve para reger a entrada de usuários em sistemas parcialmente autorizados. Usado geralmente no nível de sessão.

5.1.3 Transmita

PARÂMETROS: Arquivo Porta
Dados: um conjunto de octetos

RETORNA: menor que zero conforme o tipo de erro

Se existir a conexão e estiver aberta, envia dados ao processo par. Não aguarda a recepção da mensagem pois isto pode ser verificado por outra primitiva.

5.1.4 Receba

PARÂMETROS: Arquivo Porta
Dados: um conjunto de octetos
Limite: máximo tempo de espera

RETORNA: menor que zero se houve algum erro, a conexão não mais existe ou ocorreu estouro de tempo na espera de uma mensagem.

Obtém uma mensagem transmitida pelo processo par e a coloca em dados. Se limite for maior que zero espera tal valor em segundos, senão aguarda indefinidamente.

5.1.5 Desconecte

PARÂMETROS: Arquivo Porta

RETORNA: zero se a desconexão foi realizada corretamente.

Desconecta o processo do processo par, não aguardando, entretanto, a desconexão completar-se totalmente.

5.2 Primitivas funcionais secundárias

5.2.1 Telegrama

PARÂMETROS: Arquivo Porta
Dados: um conjunto de octetos

RETORNA: menor que zero se houve erro ou conexão não existe mais.

Transmite dados expressos ao processo par passando à frente das demais recebidas. Fica aguardando a recepção desta.

5.2.2 Ouça

PARÂMETROS: Tipo: inteiro
Dados: um conjunto de octetos

RETORNA: menor que zero se algum erro
o número da conexão de transporte.

Aguarda mensagens de vários processos par, em dados, útil quando existem diversas comunicações assíncronas e é possível então receber de diversos. Também retorna em tipo se é uma ação equivalente a RECEBA ou a CONECTE, pois também aguarda este tipo de mensagem.

5.2.3 Estado-de-conexão

PARÂMETROS: Arquivo Porta
Solicitação: texto
Retorno: texto

RETORNA: menor que zero se algum erro
o valor inteiro de algum parâmetro solicitado.

Fornece ao usuário informações gerais sobre a co-

nexão, conforme o que este solicita. Pode retornar um texto em retorno ou um valor.

As solicitações podem ser:

CÓDIGO	OBS
ESTADO	Estado da conexão - um dos descritos no capítulo anterior
CLASSE	Classe em uso do Protocolo de Transporte
EDUDPT	Se há dado expresso armazenado
MSGXMT	Mensagens transmitidas
MSGREC	Mensagens recebidas
NOMEPR	Nome do processo par
ENDPRP	Endereço do processo par

5.2.4 Altera-parâmetros

PARÂMETROS: Arquivo Porta
Tipo do parâmetro: inteiro
Valor: inteiro

RETORNA: menor que zero se houve erro.

Altera os parâmetros da conexão, do tipo temporizadores e outros. É conveniente usá-la antes de abrir a conexão.

5.3 Outras formas de acesso ao protocolo

É possível trabalhar diretamente com o Arquivo Porta, todavia será necessário conhecer profundamente o interface.

Mesmo na forma não direta é possível ainda acessar os atributos do Arquivo Porta.

6 A IMPLEMENTAÇÃO

Conforme afirmado nos capítulos anteriores a opção por máquinas Burroughs nos permite utilizar aspectos de software especialmente construídos nesta linha de equipamentos, tais como bibliotecas de primitivas funcionais de tal forma que programas em diversas linguagens possam utilizar-se delas; arquivos do tipo porta facilitando a comunicação entre programas; semáforos; tipo de dados fila (Queue); linguagens especiais para comunicação de dados e outros ...

Este capítulo pretende descrever o conjunto de rotinas que possibilitam a comunicação a nível de transporte mostrando como os recursos citados acima são utilizados.

6.1 Visão geral

Como idéia básica do projeto faz-se natural o respeito aos procedimentos sugeridos no capítulo cinco quanto à maneira de acesso aos recursos e formas de contabilização quer por um usuário qualquer quer pelo nível de sessão além da independência quanto ao nível de rede utilizado (o que na prática redundou num nível "vazio", isto é, um nível com zeros na zona de informação, conforme recomendação do Escritório Nacional de Padrões - NBS - do Departamento de Comércio Americano) /NAT 84/.

De que forma constituiria a estrutura então? Devemos considerar o fato que diversos processos estarão acessando simultaneamente um conjunto simultâneo de dados? Ou seria melhor separar e apor múltiplas cópias da entidade ao código objeto?

A solução encontrada, do ponto de vista formal, procurou ser a mais simples possível e, por isso, optamos pela criação de um *monitor* tal como descrito por Hoare e utilizado em linguagens do tipo Módula e Pascal Concorrente. Nada mais natural permitir que diversos processos aces-

sem as "rotinas de transporte" apropriando-se de dados globais somente quando estes lhe forem dispostos. Tal forma permite então a simultaneidade das diversas operações localizadas /SOU 85/.

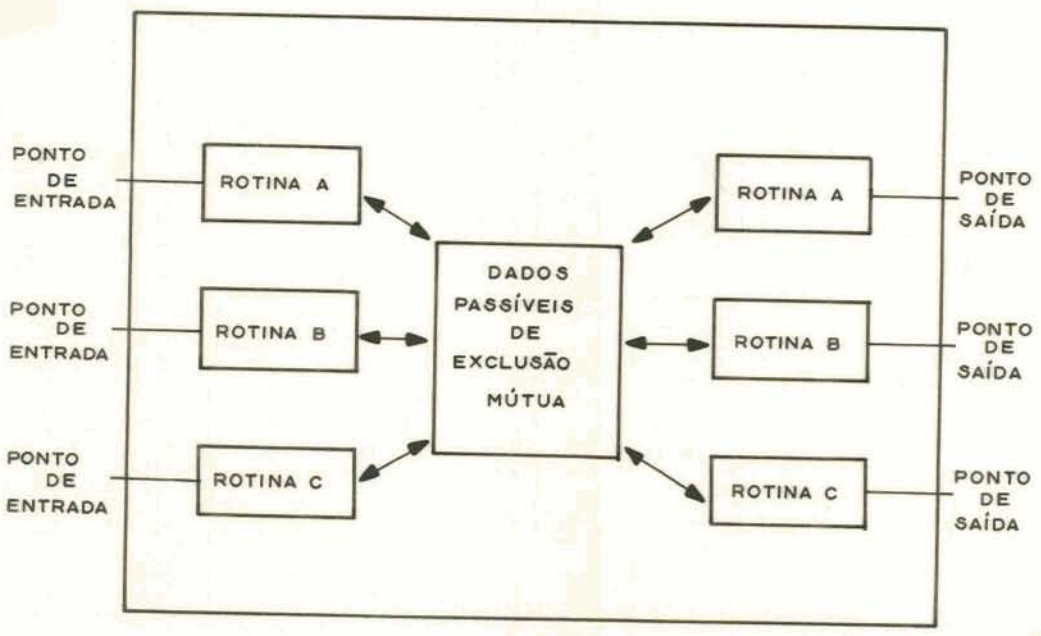


FIGURA 6.1 O monitor para o transporte

Apesar desta definição, deparamos com um problema: a linguagem utilizada, ou melhor, o software Burroughs, não apresenta monitores como uma ferramenta. A solução, então, deveria contornar este problema e assim foi feito utilizando-se a técnica de semáforos, preconizada por Djikstra e onde em DCALGOL existem os comandos PROCURE PARA P(S) e LIBERTE PARA V(S) /DEI 84/.

Desta forma, continuamos com uma visão lógica de um monitor, implantado fisicamente através de semáforos.

Com a solução acima foi permitido então definir como os diversos processos poderiam utilizar-se das rotinas de transporte. Para tanto, basta utilizar ferramentas ati-

vas, isto é, a biblioteca de rotinas em tempo de execução, descrita no capítulo 4. Passamos agora a possuir um conjunto de primitivas funcionais a serviço do usuário acessando um conjunto de dados próprios e outro conjunto de dados comuns do tipo estado da conexão, conexão livre, e outros controlados através de semáforos, todos contidos numa biblioteca /CUN 85/,/CUN 85a/ e /FUJ 84/.

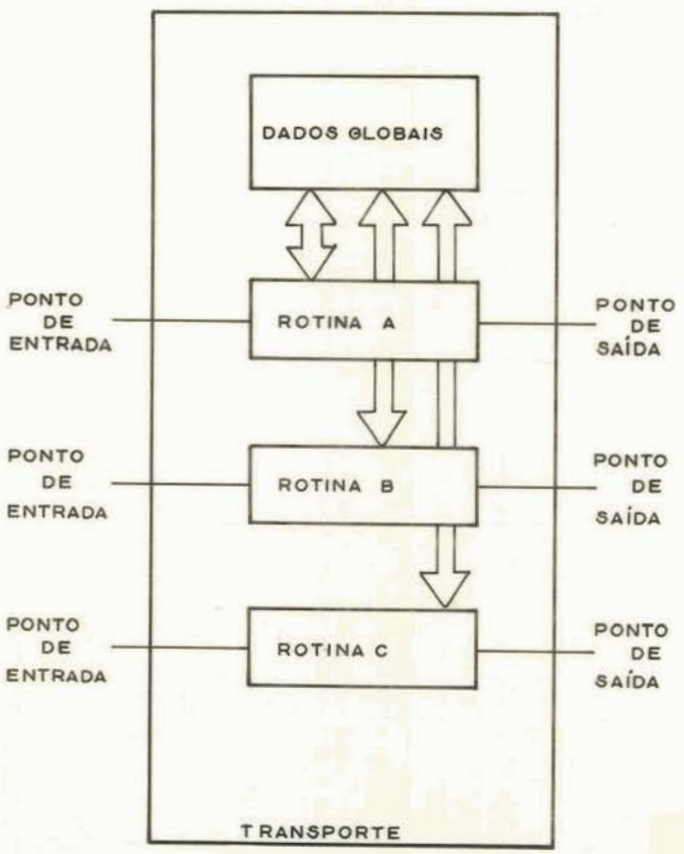


FIGURA 6.2 A biblioteca de transporte

É fácil observar que esta estrutura não basta pois as rotinas oferecidas até então não passam de mero expediente de acesso ao nível. O grosso fatalmente caberá a um processo à parte, denominado o Kernel (núcleo). Este processo consiste em receber mensagens do nível de rede, tratá-las e enviar às rotinas de acesso, receber mensagens das roti-

nas de acesso, tratá-las e enviar ao nível de rede, bem como tratar dos temporizadores de cada conexão. As duas primeiras tarefas utilizam de arquivo tipo porta para comunicação com cada rotina (ou melhor, com cada processo que está se utilizando da rotina).

Como tal processo acessa os dados globais, poderia ser também uma rotina da biblioteca. Infelizmente, uma biblioteca só contém código chamável e, portanto, não lhe é permitido ficar executando um processo interno.

A solução encontrada requereu certa engenhosidade: toda biblioteca só assim se torna quando executa o comando FREEZE, antes disto é um programa comum que é disparado quando um usuário chama uma de suas rotinas; como um programa comum antes de transformar-se numa biblioteca, dispara um outro processo assíncrono e independente que, por sua vez, chama a rotina da biblioteca que compõe-se do núcleo.

Desta forma, todo o protocolo é disparado automaticamente ou via instrução do operador através da execução em si ou via chamada do usuário ou nível superior, conforme solicitação ISO.

Assim ficou desta forma a arquitetura do sistema:

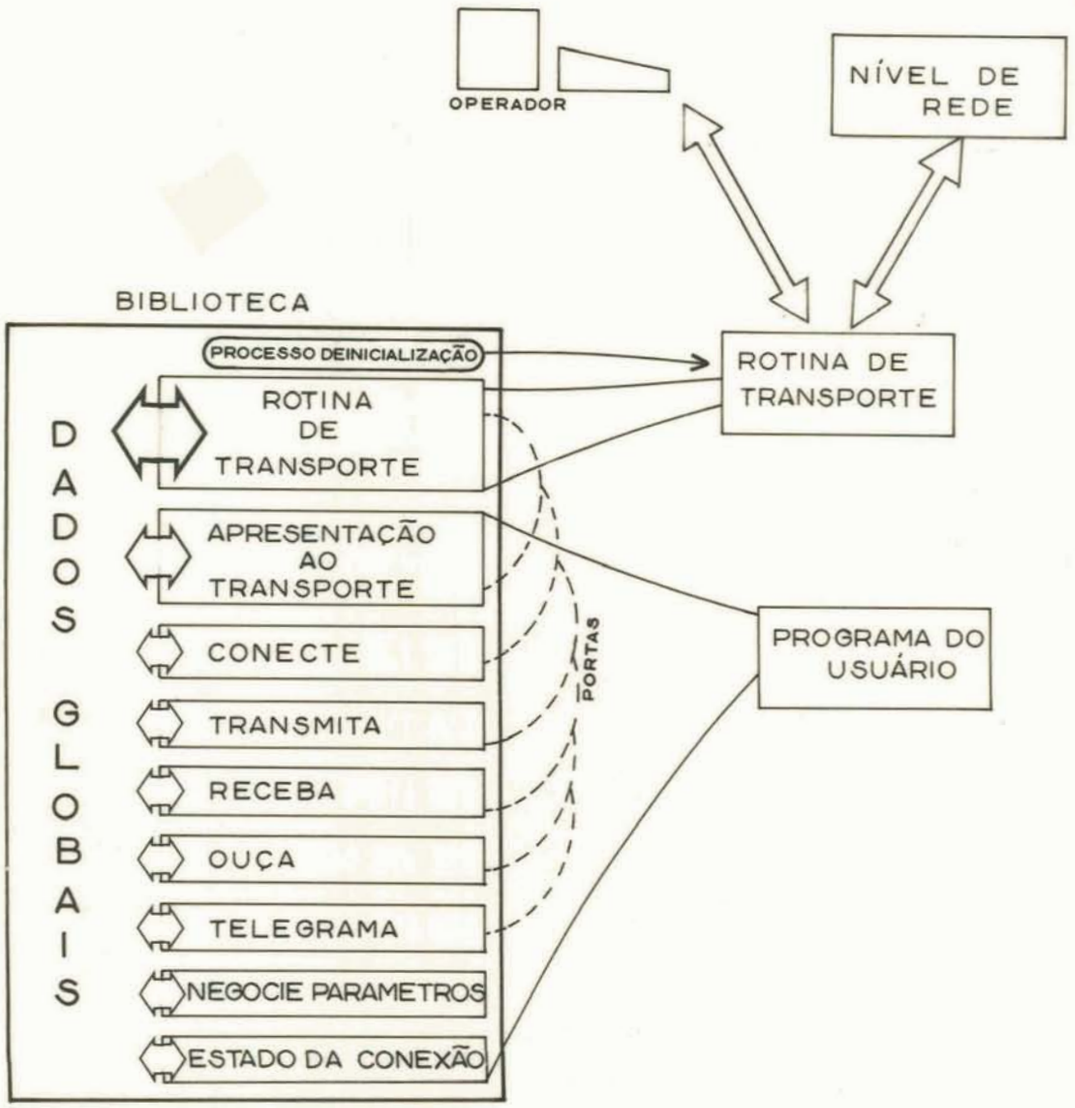


FIGURA 6.3 Visão global do sistema de transporte

6.2 Estrutura de dados

Todo sistema nada mais é que um conjunto de dados mais algoritmos; portanto, uma forma de entendê-lo consiste na apresentação dos seus dados associando-o com o trecho de programa correspondente. Não serão referidos detalhes secundários tais como aqueles que apenas facilitem a escrita. /BUR 81/

6.2.1 Semáforo-das-globais

É uma variável do tipo evento que é procurado ou liberado conforme se entra ou sai, respectivamente, de uma zona de exclusão mútua.

6.2.2 Conexões-utilizadas

É um conjunto de bits onde cada um representa a utilização de conexão. Usado em apresentação-ao-transporte e nos procedimentos de liberação de conexões. Sua utilização deve ser exclusiva.

6.2.3 Conexões-de-transporte

Cada conexão de transporte possui um registro onde estão contidos dados referentes à sua situação.

Todos estes registros são guardados em um array com o nome acima e, devido a cada conexão poder ser referenciada por um índice, conseguimos nos apropriar dos dados de uma conexão. De utilização exclusiva, serve às rotinas Negocie-Parâmetros, Status-da-Conexão, ao Kerner e basicamente a todo o sistema. Também utilizado para depuração.

Seus campos consistem de:

a) *Estado*, inteiro, octeto

Informa o estado atual da conexão, em múltiplos de 19, pois ao numerarmos eventos de zero a 18 podemos rea-

lizar o casamento estado mais evento e facilmente definir o procedimento a executar.

São treze os estados utilizados (veja o capítulo três): CLOSED, WFNG, WBCL, WFCC, WFTRESP, AKWAIT, OPENED, CLOSING, REFWAIT, WFCC-R, OPEN-R, WFEA e SCLOSEDW.

b) *Referência-par*, inteiro, octeto

Contém o índice do par da conexão. Se não houver será zero. Utilizado nas UDPTs no campo referência-par.

c) *Classe-do-protocolo*, inteiro, 3 bits

Contém a classe do protocolo a ser utilizada.Classes válidas: zero, dois e quatro.

d) *Contador-de-retransmissões*, inteiro, octeto

Quando uma UDPT de dados é enviada, a entidade par deve responder com uma UDPT de reconhecimento (AK). Se após um determinado período isto não ocorrer, ocorre uma retransmissão. Este procedimento se repete até que atinja um número máximo. Esta variável contém este valor.

e) *TPDU-ED-Armazenada*, lógico
HÁ-ED-TPDU-Armazenada, lógico

A primeira informa se a conexão recebeu algum dado expresso, que deve ser atendido prioritariamente. A segunda, se a conexão possui algum dado expresso a transmitir, o que também deve ser realizado prioritariamente, salvo uma liberação de conexão.

f) *Tamanho-da-janela*, inteiro, três bits

Informa o tamanho da janela de transmissão para efeitos de controle de fluxo. O valor por omissão é 8,tendo como limite 128. Negociado na abertura da conexão.

g) *Versão*, inteiro, octeto

A versão do protocolo implementado. Atualmente é um.

h) *TPDU-esperando-confirmação*, lógico

Informa se há alguma UDPT aguardando confirmação. Utilizado em conjunto com o controle de retransmissões.

i) *Endereço-de-rede*, inteiro, duplo octeto
Endereço-de-rede-do-par, inteiro, duplo octeto

O primeiro contém o endereço da máquina em que está contido o protocolo de transporte. O segundo refere-se ao processo par. Utilizados para comunicação com o nível de rede.

j) *Nro-do-meu-processo*, inteiro, octeto duplo

Cada processo na linha Burroughs utiliza uma identificação numérica de 1 a 9999 chamada *Mix Number*. Este número é utilizado para associar o nome do arquivo porta à conexão em questão.

l) *Fui-eu-quem-iniciou-a-conexão*, lógico

Informa qual dos dois disparou os procedimentos de abertura de conexão. É utilizado nas rotinas Conecte e Ouça.

m) *Credito*, inteiro, octeto

Utilizado em conjunto com tamanho-da-janela para controle de fluxo. Informa até quantas UDPT's a conexão pode ainda enviar sem esperar por um AK. Este campo pode ser recebido na UDPT de dados ou decrementado a cada transmissão.

n) *Cont-TPDUs-recebidas*, inteiro, octeto

Uma mensagem pode conter mais de uma UDPT. Esta variável é um contador para sabermos quantas UDPTs foram recebidas e não montadas para envio ao nível superior. Veja o próximo item.

o) *Índice-dado-inicial*, inteiro, octeto
Índice-dado-final, inteiro, octeto

Usados em conjunto com o item anterior define exatamente quais UDPTs serão usadas na montagem para enviar ao nível superior a mensagem completa.

Não podemos esquecer que as UDPTs podem vir fora da ordem enviada, porta as três últimas variáveis citadas nos ajudam a compor a mensagem corretamente.

p) *Prox-seq-a-transmitir*, inteiro, octeto
Prox-seq-a-receber, inteiro, octeto

As UDPTs de dados são numeradas através de uma seqüência (como o protocolo só trabalha na forma normal, módulo 256), estas variáveis informam qual a próxima seqüência da UDPT deve ser recebida ou transmitida.

De grande importância na transmissão, na recepção, devido ao fato das UDPTs poderem chegar fora da ordem transmitida, serve apenas para indicação de que o protocolo deve reconhecer a mensagem; após, será reconhecida a última recebida desde que não forme um "buraco".

q) *Prox-seq-ed-a-transmitir*, inteiro, octeto
Prox-seq-ed-a-receber, inteiro, octeto

O mesmo que o item anterior, contudo relacionados com dados expressos. Como não há controle de fluxo e os dados expressos são reconhecidos um a um aqui se tem exatamente a seqüência esperada.

r) *Max-seq-a-transmitir*, inteiro, octeto

Em conjunto com crédito e/ou tamanho-da-janela acrescido de prox-seq-a-transmitir forma o espectro de variáveis que realizam o controle de fluxo do protocolo de transporte. Max-seq-a-transmitir define qual o número máximo de seqüência que o protocolo pode utilizar e, por conseguinte, quantas UDPTs podem ser liberadas.

A fórmula para obtenção desta variável é:

MAX-SEQ-A-TRANSMITIR = PROX-SEQ-A-TRANSMITIR +
+ MIN(CREDITO,TAMANHO-DA-JANELA)
- 1.

s) *Max-seq-a-receber*, inteiro, octeto

Vale prox-seq-a-receber + tamanho-da-janela - 1. Indica qual a variação de números de seqüência é aceita como válida na recepção de UDPTs de dados. Todas as demais, que estiverem fora do intervalo fechado prox-seq-a-receber e max-seq-a-receber serão sumariamente ignoradas (pode ser uma repetição por perda de um AK, se não o for será retransmitida).

6.2.3.1 Nomes de processos

Associados com cada registro de conexões de transporte, controlado pelo mesmo índice ainda existem dois *strings* para identificação dos processos que controlam a conexão:

a) Nome-do-processo: o nome do possuidor da referida conexão de transporte, obtido em apresentação-ao-transporte;

b) Nome-do-processo-par: o nome do possuidor da conexão par, obtido na fase de abertura de conexão.

Ambos os nomes podem possuir, no máximo, 17 caraç

teres.

6.2.3.2 Segurança

O *string* segurança é o código do usuário que deve o processo par casar num pedido de conexão para que esta solicitação não seja negada por violação de segurança. Obtido em apresentação-ao-transporte para posterior casamento, corresponde na linha Burroughs ao atributo USERCODE.

É de implementação opcional, conforme preparação na instalação e, se for utilizado, deve conter no máximo 17 caracteres.

No atendimento ao pedido de conexão, o casamento não só é feito através do código de segurança mas do mecanismo completo USERCODE/PASSWORD, isto é, na linha Burroughs cada USERCODE possui uma senha secreta controlada pelo usuário. O prendedor do transporte deve realizar então uma validação através de primitivas do sistema operacional chamadas USERDATA. Eis mais um motivo de utilizar Algol na sua extensão DCAlgol, a única linguagem em que nos é permitido realizar tal feito.

6.2.4 Definições dos eventos

Cada evento é tratado como uma constante de valor numérico, de 1 a 17, é uma sinalização se está "chegando ou saindo" (se vem do nível de rede o procedimento ou do nível de sessão). Após a obtenção do evento é feita sempre uma operação chamada evento-versus-estado que indicará na tabela de estados qual a grade, e o procedimento a realizar:

EVENTO-VERSUS-ESTADO = ESTADO + EVENTO
(ESTADO: MÚLTIPLO DE 17; EVENTO: DE 1 A 17)

6.2.5 Campos para comunicação com o nível superior

A comunicação com o nível de sessão ou o próprio usuário, como já dito anteriormente, é realizada através de arquivos do tipo porta.

Além do mais, é natural que haja uma regra de composição da mensagem, isto feito aproveitando-se o fato das máquinas Burroughs utilizarem palavras reais de 6 octetos. Eis a formatação dos campos:

a) Octetos 0 a 2 - Tipo da mensagem, isto é, seu código de requisição ou indicação.

a') *Requisição* (do usuário ao provedor)

3 - requisição de conexão
4 - resposta à indicação de conexão
5 - requisição de transmissão de dados
9 - requisição de transmissão de dados expressos
10 - requisição de desconexão

a") *Indicação* (do provedor ao usuário)

1 - indicação da chegada de um pedido de conexão
2 - confirmação de uma conexão
3 - dados chegando
4 - dados expressos chegando
5 - indicação de término de conexão
6 - confirmação a um pedido de encerramento de conexão.

b) Octetos 3 a 5 - Número da conexão de transporte.

c) Palavra 1 - Endereço de rede da conexão par.

d) Palavras 2 a 4 - Nome do processo par.

e) Palavras 3 em diante - Dados da mensagem, expressos ou não.

f) Palavras 5 a 7 - Código de segurança/opcional.

Os itens d e f são utilizados apenas nos códigos 3 e 4 para requisição e 1 e 2 para indicação passando a parte de dados para a palavra 7.

6.2.6 Campos para comunicação com o nível inferior

A comunicação com o nível de rede se utiliza das unidades de dados do serviço de rede (UDSR) conforme receita o padrão ISO. No caso especial da implementação do nosso nível de rede este é vazio, todavia o provedor de transporte está preparado para outros níveis de rede, como o X.25, breve lançamento na linha Burroughs.

Seus campos compõem-se de:

a) Tipo, octeto, tipo da unidade de dados:

0 - indicação de chegada de dados ou requisição de transmissão de dados (lembremos que dados referentes ao *nível de rede*);

7 - requisição de desconexão de rede;

8 - requisição de uma conexão de rede;

11 - indicação de desconexão de rede;

16 - confirmação a uma solicitação de conexão de rede;

17 - indicação de um "reset" na conexão de rede.

b) Endereço do chamado, octeto duplo:

Endereço de rede à qual pertence o processo com

que o processo responsável pela conexão em questão deseja comunicar-se.

c) Endereço do chamante, octeto duplo:

Endereço de rede do sistema, ou da conexão, a que estão referidas *todas* as UDPTs contidas nesta UDSR.

d) Qualidade do serviço:

Define os padrões de performance e qualidade desejáveis ao nível de rede. Nesta implementação é sempre zero.

e) Dados.

6.2.7 Arquivos utilizados para comunicação

Existem dois arquivos, ou melhor, conjuntos de arquivos, utilizados para comunicação.

6.2.7.1 Arquivo *interface*

O arquivo *interface* é o interface para o nível superior, um arquivo de portas onde os processos devem inserir suas comunicações e instruções à entidade de transporte.

Conforme descrito no capítulo 4 que possui sub-arquivos no total equivalente ao número máximo de conexões de transporte simultâneas (por omissão o máximo de conexões está em 25 mas é possível, na implantação do sistema, chegar a 255).

É interessante detalhar alguns atributos desse arquivo:

a) Title: "COMMSTRANSPORTE";
b) Myname: "ISOTRANSPORTE".

O casamento é realizado em apresentação ao trans-

porte através das sub-portas (uma para cada conexão) onde o atributo YOURNAME torna-se: "TP", 4 dígitos contendo o número de mix do processo possuidor da conexão, o número da conexão e "transporte".

TP mix<do processo><índice da conexão>TRANSPORTE.

Exemplo: TP969704TRANSPORTE.

Isto garante a unicidade do subarquivo. Para efeitos de compatibilização é aposto o código de segurança(USER CODE) do processo possuidor da conexão a cada sub-arquivo porta.

6.2.7.2 Arquivo rede

Há duas formas de tratar esta comunicação conforme o nível de rede. Se esse for um MCS que utilize arquivos remotos temos o DIRECT FILE se aceitar portas existe um arquivo tipo porta.

6.2.7.2.1 Arquivo de rede remoto

O problema de utilizarmos o arquivo remoto puro e simples (isto é, usando o subsistema de entrada e saída padrão do MCP) é que necessitaríamos ficar constantemente inquirindo se chegou alguma mensagem para após ler ou,pior, adormecer todo o processo até que venha uma mensagem, isto seria uma completa insensatez.

Em DCALGOL é permitido a utilização de entrada e saída sem os recursos dos métodos de acesso, causando um pouco mais de trabalho e muito mais cuidado mas liberando-nos para tarefas que destoam das aplicações normais. No caso, existe o arquivo DIRECT e, particularmente, DIRECT REMOTE FILE, onde todo o tratamento é realizado a mão.

Estamos nos referindo a um arquivo remoto (de comunicação de dados) onde existe um evento associado em que

é permitido a um processo adormecer por um conjunto deste tipo e mais outros e acordar quando um deles for sinalizado (isto se dá através de procedimentos do tipo BLOCK e WAKEUP descritos por Holt /DEI 84/).

Outra vantagem deste tipo de arquivo é o assincronismo, tanto na leitura, quanto na gravação. Isto permite ao processo gravar enquanto faz outras atividades. Veremos como:

```
          boolean leitura-bloqueada;
            :
          leitura-bloqueada:= false;
            :
          loop
            :
(1)       it not leitura bloqueada
          then begin
               read (rede,buffee-da-rede)[evento-rede]
               leitura-bloqueada:= true;
               end;

(2)       wait (temporizadores,
                outros-eventos (interfaces superiores,dados
                                expressos, operador)
                evento-da-rede);

(3)       if happened (evento-da-rede)
          then begin
               reset (evento-da-rede)
             → o texto já está em buffer de rede ←
               leitura-bloqueada:= false;
               tratamento-da-rede;
               end;

          end loop;
                :
```

```
          write (rede, buffer) [evento-rede]
          → outros procedimentos ←
(4)       wait (evento-rede)
```

Em (1) só podemos realizar a leitura se não a fizemos ainda pois, se isto ocorrer um buffer será sobreposto ao anterior.

Em (2) todo o processo adormece pois não há o que fazer. Quando um evento modificar o estado do processo ele o atende (3) (pode ser até um intervalo mínimo de tempo).Se chegou alguma mensagem da rede esta será atendida imediatamente acordando o processo. Se chegarem mais de uma simultaneamente, as demais serão enfileiradas, contudo poderão ser "lidas" através da instrução READ para serem postas em Buffer-da-rede.

Em (4) há a gravação que pode ser feita no início de um conjunto de procedimentos independentes da realização de fato da operação de entrada e saída e após, aí sim,aguardar a liberação do Buffer (o que normalmente já ocorreu).

6.2.7.2.2 Arquivo de rede tipo porta

Possui as mesmas vantagens do anterior, todavia de mais simples utilização e, como contratempo, de mais performance inferior por utilizar-se de chamadas mais complexas ao sistema operacional.

Através da instrução DEFINE do Algol permitimo-nos utilizar o mesmo código fonte para declarações diferentes. É bom lembrar que a implementação permite apenas uma das maneiras.

Para teste numa mesma máquina este foi o tipo de arquivo utilizado. Os motivos serão vistos na seção 7.1.1.

6.2.8 Interface para dados expressos

Devido ao fato de ser expressa esta função possui um tratamento privilegiado, isto é, um atendimento prioritário aos dados comuns. Para tanto, a manipulação é diretamente realizada por variáveis globais: um conjunto de eventos, TPDUs-expressas-EU, sendo um para cada conexão e uma fila canal-expresso onde são apostas as mensagens expressas.

A fila é formada através do tipo de dados Queue do DCAlgol onde as mensagens podem ser removidas e retiradas na forma Primeiro a entrar - Primeiro a sair. À fila está associado um evento no qual o processo pode esperar a existência de uma mensagem /BUR 84d/ e /BUR 80/.

Como não é permitido mais de uma mensagem de dados expressos por vez quando, ao ser removida a mensagem da fila, o processo a atender é causado o evento TPDU-EXPRESSAS-EV da conexão para que esta possa voltar aos seus outros procedimentos.

6.3 Procedimentos de apoio

Tais procedimentos são utilizados apenas pela rotina de transporte e se separássemos como foi feito com o resto em estruturas de dados estanques dos algoritmos não teríamos a idéia exata de seu relacionamento.

Sendo assim, o melhor é apresentá-las junto com os procedimentos associados de maneira a ser possível o entendimento de suas estruturas.

6.3.1 TPDU

O array TPDU é uma imagem da UDPT trabalhada. Utilizando-se de uma possibilidade, em DCALGOL, de definir estruturas distintas no mesmo endereço, relativo, de memória, esta estrutura pode ser um conjunto de octetos, de campos

lógicos ou de palavras de 48 bits.

Através da instrução DEFINE se torna possível utilizar uma única estrutura para todas as UDPTs, visto serem manipuladas uma por vez.

Além, então, das definições dos campos na forma

```
LI     = CT(1, OCTETO)
CODIGO = CT(2, OCTETO)
CODE   = CT(2, MEIO OCTETO)
CDT    = CT(2.5, MEIO OCTETO)    etc.
```

existem dois procedimentos que permitem a fácil manipulação:

a) CT(OCTETO,TAMANHO) - declara um campo a partir de octeto com a extensão de tamanho, que pode ser octeto, octeto duplo, bit ou meio octeto;

b) PREENCHA(INDICE,ITEM) - coloca o conteúdo de item no octeto discriminado em índice.

6.3.2 Apoio à transmissão dos dados

Há uma lista de UDPTs transmitidas para cada conexão de transporte contendo o número de seqüência da UDPT e uma bit informando se está aguardando AK ou não.

Além disto há uma lista de dados transmitidos para cada conexão divididos em grupo de octetos com tamanho equivalente ao campo de dados da UDPT DT assegurando que através do número de seqüência, obtido na lista anterior e firmado seu índice, podemos por um cálculo simples reobter a UDPT se for necessária uma transmissão. Cumpre lembrar que, se esta UDPT não estiver esperando AK, ela pode ser descartada.

Como, ao receber uma mensagem para transmissão, esta é logo desmontada em partes iguais, exceto a última se

menor, e o limite de UDPTs aguardando reconhecimento é dado pelo tamanho da janela atual, é necessário ainda armazenar o resto da unidade de serviço. Isto é feito através de um array de textos e um array com contadores de UDPTs, ambos para cada conexão de transporte.

Como um adendo, é de bom tom comentar a existência de um operador dentro do conjunto de instruções das máquinas de grande porte Burroughs, chamado MASKSEARCH que pesquisa num array um campo, determinado por uma máscara, atrás de um valor pré-especificado. Se isto se suceder em campos de até uma palavra estaremos fazendo pesquisas com apenas uma instrução de máquina. Este artifício é largamente utilizado não só aqui mas onde for necessário constante pesquisa. Daí o fato da fragmentação um tanto artificial dos dados.

Isto posto, torna-se possível escrever as rotinas que irão apoiar a transmissão.

6.3.2.1 Obtém lugar para inserir TPDU

Coloca a UDPT na lista de UDTPs transmitidas procurando um espaço disponível através da pesquisa na lista de UDPTs transmitidas por uma UDPT que não mais esteja aguardando reconhecimento.

6.3.2.2 Obtém texto

Através do número de seqüência, obtém os dados contidos na UDPT em questão. Faz isto através da correspondência entre a lista de UDPTs transmitidas e a de dados transmitidos.

6.3.2.3 Construa e armazene TPDU

Obtém da unidade de dados do serviço de transporte transmitida pelo usuário a mensagem que esta intenta en-

viar. Após segmenta-a em pacotes iguais, de tamanho equivalente à área de dados da UDPT DT, e concatena no array de textos mencionado no item 6.3.2, TPDUs-armazenadas, apondo um código para informar se este pacote é ou não o último.

6.3.2.4 Transmita TPDUs armazenadas até limite janela

Enquanto houver crédito ou ainda UDPTs a transmitir (estes limites são obtidos quando prox-seq-a-transmitir for igual a max-seq-a-transmitir ou o contador de UDPTs armazenadas for zero, respectivamente) obtém o texto armazenado e elimina-o desta estrutura.

Após, e com um novo número de seqüência, obtém lugar para inserir UDPT, guardando-a para posterior reconhecimento, não sem antes preencher os campos de cabeçalho.

Por fim monta a UDSR e transmite à rede.

6.3.2.5 Retransmissão

A rotina verifica quais as UDPTs estão aguardando reconhecimento através da lista de UDPTs transmitidas e retransmite todas, incrementando o contador de retransmissões.Se este atingiu o máximo, então a conexão é encerrada.

É bom lembrar que jamais haverá um número maior de UDPTs aguardando que o da janela.

6.3.3 Apoio à recepção de dados

Há um conjunto de octetos contendo o texto recebido por conexão, o texto completo enviado pelo processo par na UDST da entidade par. Esse texto vai sendo montado conforme chegam as UDPTs através de concatenação, até completar o texto.

6.3.3.1 Insere mensagem recebida

É a rotina que primordialmente realiza a concatenação dos dados recebidos nas UDPTs DT ao array texto recebido da conexão.

Todavia, as UDPTs podem não chegar na ordem tal como enviada, portanto a concatenação não será apenas uma a posição ao fim do texto e sim uma obtenção deste e através da fórmula

(SEQÜÊNCIA-DA-UDPT - INDICE-DADO-INICIAL) x TAMANHO-DADOS-UDPT

podemos obter o início de tais dados no texto, abrindo-o e inserindo-o.

Quando o contador de UDPTs recebidas for igual à diferença entre índice-dado-inicial e índice-dado-final,significa que todas as UDPTs referentes à mensagem contida na UDST foram recebidas e portanto deve ser enviado ao usuário (ver rotina INFORME).

6.3.4 Dados expressos

A capacidade de tratar dados expressos é muito parecida com dados comuns, mas, como não devemos possuir múltiplas UDPTs, a tarefa é bem mais simples.

Para cada conexão há um texto onde é guardada a UDPT já transmitida. Antes disso, a UDST que requer o serviço não é passada via porta e sim por uma variável do tipo QUEVE (fila). Tal indicará à rotina de transmissão a ocorrência e esta a tratará prioritariamente, transmitindo.

Para a recepção, nada é guardado, ao contrário, busca-se a transmissão o mais rápido possível, isto é, imediatamente após a chegada dos dados.

É importante frisar que não há uma prioridade em relação aos demais processos e sim aos dados da própria co-

nexão e, portanto, não é necessário uma contabilização especial.

6.3.5 Controle de temporizadores

Estas rotinas são muito importantes e como são processadas em tempo real devem ser realizadas da maneira mais rápida possível.

Um recurso para otimizar estas operações está numa instrução de máquina, LLLU-LINKED LIST LOOK UP, que faz pesquisas numa lista encadeada até que um campo seja maior que um parâmetro.

À parte isso, segundo permissão do padrão, optou-se por utilizar um temporizador apenas para toda a janela. Desta forma, teremos quatro temporizadores:

a) T-REF, temporizador para referências congeladas;

b) T-WINDOW, temporizador para informar o estouro de tempo, sem atualização da janela e que, portanto, é mister enviar um AK a esmo;

c) T-RETRANS, temporizador para indicar que estourou o período para aguardo de reconhecimento;

d) T-INACT, calcula o tempo de inatividade que, se expirar, força o encerramento da conexão.

Como as pesquisas devem ser feitas em todos os temporizadores não é possível subdividi-los em conjuntos por conexão, todavia, temos que, de alguma forma, recuperar o índice que nos dará a conexão referente.

Há, então, uma lista de temporizadores organizada da seguinte forma:

1. os quatro temporizadores, na ordem acima, pos-

tos em linha, por conexão - logo a obtenção do índice da conexão é obtido através da divisão inteira do índice do temporizador e o temporizador pelo módulo desta;

2. um ponteiro que aponta para o próximo temporizador, existente por causa do operador LLLU;

3. um campo de valor contendo o temporizador.

Quando valor for zero, um temporizador estará inativo e o início dos tempos é marcado com o valor $2^{28}-1$, limitado pelo campo de valor (e o que limita a execução ininterrupta do programa em 310 dias) e decrementado em décimos de segundo. Temos, então, algumas rotinas especiais.

6.3.5.1 Inicialização de temporizadores

Monta a lista com os índices para posterior utilização do LLLU.

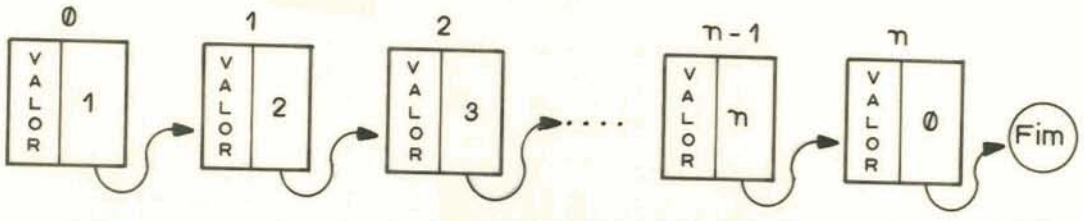


FIGURA 6.4 Organização da lista de temporizadores

Executada apenas no início da rotina de transporte.

6.3.5.2 Insere temporizadores em nova conexão

É a rotina que, quando uma conexão sai do estado CLOSED, insere os temporizadores na lista.

Esta inserção significa transformar os valores dos

temporizadores marcados pelo índice como ativos (diferentes de Ø) e acerta os ponteiros. Na inicialização, embora marque-se os ponteiros, o primeiro dirige-se ao último, para otimização e, aqui, então há os apontamentos corretos.

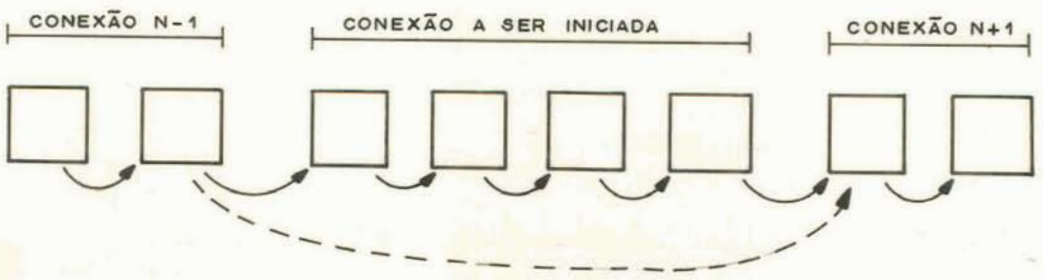


FIGURA 6.5 Inserção lógica de temporizadores
Basta modificar o ponteiro final da conexão N-1, que apontava para o inicial da NM, conforme linha pontilhada.

6.3.5.3 Retira temporizadores

Marca os temporizadores como inativos e modifica o último ponteiro da conexão anterior para o próximo, otimizando a procura.

6.3.5.4 Procure timeout

Recebe como parâmetro uma variável contendo a hora do dia na forma

$2^{28}-1$ - (Tempo de duração do programa em décimos de segundo)

e, utilizando-se da instrução LLLU, retorna o índice do primeiro temporizador, não importa qual conexão, cujo valor indica que seu limite é menor que a hora em questão.

6.3.6 Ações

Segundo a norma ISO foram definidas nove ações padrão:

1) ativa T-REF;

2) se verdadeira ativa T-WINDOW, senão desativa;

3) se verdadeira ativa T-INACT, do contrário, desativa;

4) se verdadeira, incrementa contador de retransmissões e ativa seu temporizador, senão, torna-o inativo e zera o contador;

5) reduz crédito;

6) atualiza crédito;

7) transmite UDPTs, armazenadas até o limite permitido pela janela (item 6.3.2.4);

8) armazenar os temporizadores - não foi encontrada utilidade na implementação;

9) torne T-INACT inativo (pois é zerado a cada menor sinal de inatividade).

6.3.7 Predicados

Conforme a norma ISO, para a execução do diagrama de estados, tanto para efeitos de correção ou de tomada de decisão são fornecidos dez predicados, ditos por:

1) UDPT CR aceitável;

2) UDPT CC aceitável;

3) faz-se necessário uma conexão de rede nova;

4) contador de retransmissões ainda não ultrapas-

sou o limite;

5) há UDPT ED armazenada;

6) o campo EOT da UDPT DT (dados) está sem um, isto é, a UDPT que trouxe o último segmento de uma mensagem;

7) crédito igual a zero (janela fechada);

8) há uma UDPT ED esperando para ser transmitida;

9) a UDPT DT está dentro da janela de recepção;

10) a inicialização da conexão de rede foi devida a esta conexão de transporte.

6.3.8 Transmissão de uma UDPT isolada

Realizada através da rotina transmitida, é montado o cabeçalho deixando espaço para CHECKSUM, dados e somente no final são calculados os dígitos de controle e apostos no texto conforme algoritmo do capítulo 3.

A seguir monta a UDSR e transmite ao nível anterior.

Um caso especial a ser tratado é a transmissão da UDPT CR onde o cabeçalho é todo especial. Além do fixo é enviado o nome do processo no campo de ponto de acesso ao serviço de transporte (PAST) chamante, o nome do processo par desejado como PAST chamado, tamanho da UDPT, a versão e o código de segurança se implementado.

6.3.9 Envio de unidade de serviço ao nível superior

Realizado através da rotina informe, ela monta a UDST conforme parâmetro e se o interface estiver disponível transmite através da subporta correspondente.

A transmissão é feita assincronamente, isto é,

não se aguarda a recepção no outro lado.

6.3.10 Encerramento da conexão

Após os procedimentos normais previstos no padrão esta rotina deve fechar o arquivo interface correspondente à conexão. Isto causará uma sinalização no evento correspondente e a rotina de transporte também, por conseguinte, fechará o seu par, limpando qualquer sobra de mensagem.

Após limpa-se os temporizadores procurando otimizar performance e por fim colocando em conexões utilizado o bit equivalente à conexão como zero, isto é, disponível a outra.

Veja que os processos de limpeza nas duas rotinas podem ser realizados até simultaneamente. Este tipo de atividade permite que um pequeno número possível de conexões possa suportar uma demanda razoável para uma grande máquina.

6.3.11 Gerenciamento da camada de rede

É a rotina que trata dos eventos causados pelo nível de rede a uma determinada conexão de transporte, já identificada anteriormente.

Se o evento for uma indicação de desconexão de rede podemos tomar as seguintes atitudes, dependendo do estado:

a) WFNC - informar o encerramento ao nível superior e passar o estado para CLOSED;

b) CLOSING

WBCL - passar para o estado REFWAIT esperando um período por mensagens mais lentas;

c) WFCC - requisitar nova conexão de rede e pas-

sar para o estado WFCC-R;

d) WPTRESP

AK WAIT

OPEN - Se foi a conexão que tomou a iniciativa então requisitar nova conexão, salvar os registradores e passar para OPEN-R, senão repetir item b;

e) WFCC-R

OPEN-R - Enquanto não for atingido o número máximo de retransmissões deve tentar requisitando novas conexões, senão encerrar como item b.

Se o evento for a confirmação a uma solicitação de conexão de rede, transmita-se uma UDPT CR e passe a aguardar uma UDPT CC.

Um reset deixa a conexão indiferente.

6.3.12 Validação de uma UDPT

Uma UDPT é válida quando todos os parâmetros indicados estão conforme os padrões definidos no protocolo. É aceitável quando a referência aponta para uma conexão existente, se não for uma UDPT CR.

Ainda, seu checksum deve estar correto e seus campos de seqüência, se houver, devem estar dentro da janela de recepção.

6.3.13 A variável TC

O inteiro mais importante de todo o sistema. Logo que é descoberto o índice da conexão este é passado para TC onde todas as listas e demais escrituras de dados que contenham informações específicas sobre conexões de transporte se utilizam para uma referência indexada.

O efeito torna-se semelhante ao WITH do Pascal:

```
WITH <conexão [I]>
DO ...
   ...
END;
```

6.4 Rotinas de interface com o nível superior

Embora já explicada sua funcionalidade é útil um detalhamento maior destas tão importantes rotinas. Isto facilitará a compreensão do paralelismo e dos procedimentos contabilizáveis ao processo, isto é, as próprias rotinas de interface.

Todas estas rotinas são exportadas e, por isso,podem ser utilizadas em qualquer linguagem.

Antes da leitura desta seção é imperativo o estudo detalhado da seção 5.1, integralmente, pois é lá onde estão as definições destes processos.

6.4.1 Apresentação-ao-transporte

Apresentação-ao-transporte procura uma conexão livre (existe um operador - Firstone - que procura o primeiro bit setado numa palavra, sua negação procura um zero).Se não houver retorna um valor negativo.

Após, liga o bit e marca o estado da conexão com índice equivalente como CLOSED, além de colocar valores em algumas variáveis de conexões de transporte.

Trabalhando com o arquivo parâmetro, transforma o atributo TITLE do arquivo no mesmo do arquivo interface. Outros atributos tornam-se:

YOURNAME - ISOTRANSPORTE

MYNAME - TPMMMMCCCTRANSPORTE onde
MMMM = número do processo
CCC = número da conexão

USERCODE - o do processo.

Ainda coloca o nome do processo na lista própria assim como o código do usuário.

Por fim, abre a subporta equivalente do interface e o arquivo parâmetro, ambos assincronamente de tal forma que o casamento seja sentido na rotina de transporte.

6.4.2 Conecte

Se o texto referente ao nome do processo par for nulo a rotina deve ficar aguardando uma indicação de conexão. Quando isto ocorrer, e o código de segurança for o correto (se exigido) será enviado ao servidor de transporte uma UDST TCONRESP aceitando a conexão.

Senão, além de obter o nome no processo par desejado obtém o endereço de rede e, se for o caso, o código de segurança. Após, transmite uma UDST TCONREQ e se o usuário desejar esperar retorno aguarda uma UDST TCONIND contra respondendo com uma TCONRESP.

Retorna o índice da conexão.

6.4.3 Transmita

Um detalhe importante a ser cuidado nesta fase é que pode não mais ocorrer a existência da conexão e, portanto, o usuário deve ser avisado.

Logo, se após testar o estado da porta, esta estiver íntegra, TRANSMITA monta a mensagem e envia uma UDST TDTREQ, sem esperar confirmação pela recepção.

6.4.4 Receba

Para maior eficiência, testa o atributo CENSUS da porta para verificar se já não chegou alguma mensagem que está apenas aguardando ser retirada. Se assim o for retira-a.

Se o usuário não especificou limite de tempo aguarda indefinidamente até receber uma mensagem.

Nos casos de haver uma mensagem esta UDST deve ser TDTIND ou TEXIND onde, então, será montado no string dados o texto desejado. Não informado se a UDST foi expressa ou não.

Se receber qualquer outro tipo de UDST seu tipo, maior que zero, será retornado junto com todo o campo de dados.

6.4.5 Desconecte

Se a porta ainda estiver aberta então envia a UDST TDISREQ e fecha assincronamente o arquivo.

RETORNA -1 se a porta não estiver íntegra.

6.4.6 Telegrama

Dados expressos não podem ultrapassar 16 octetos, nesta implementação; logo, se isto ocorrer, haverá uma rejeição por parte da rotina.

Após teste de integridade da porta para assegurar-se da também integridade da conexão, pesquisa no seu TITLE (nome) o índice da conexão de transporte do usuário. Com isto monta a UDST TEXREQ e, ao invés de transmitir pela porta, insere na fila canal expresso passando à frente das demais.

Após fica aguardando que o evento TDPU - expressa-atendida seja causado para assim liberar-se dando certeza ao usuário que a mensagem já chegou ao destino quando este retomar o controle do processo.

6.4.7 Ouça

Neste caso o arquivo porta deve, na realidade, compor-se de um conjunto de sub-arquivos.

Esta rotina fica aguardando que ocorra um dos seguintes eventos:

a) INPUTEVENT na porta - alguma mensagem chegou;

b) CHANGEEVENT na porta - algo ocorreu em uma conexão, uma abertura ou encerramento.

Após verificar, pelo índice do evento qual a conexão - veja que se não houver, Apresentação-ao-transporte será chamada para o sub-arquivo equivalente - coloca em tipo e envia a mensagem.

6.4.8 Estado-da-conexão

Através do nome do arquivo porta identifica o índice da conexão de transporte. Com este, poderá acessar o estado da conexão e seus registros referenciando a conexão correta.

Se o parâmetro desejado estiver correto, procura no campo em questão enviando ao usuário no texto ou no inteiro conforme o tipo do dado.

6.4.8 Altera-parâmetros

Com o arquivo porta descobre o índice da conexão e, se o parâmetro for válido, verifica se este pode ser negociado no tempo em questão, isto é, se a qualquer tempo ou apenas no estado CLOSED, e seus valores. Retorna zero se aceitou a alteração.

6.5 A rotina de transporte

A rotina de transporte é o cerne do sistema implementado. É por ela que trafegam as mensagens vindas do nível de rede bem como as passadas pelas rotinas que comunicam-se com o usuário.

A rotina de transporte opera ainda com a temporização, o redirecionamento, montahem, segmentação da mensagem e o faz segundo o diagrama de estados definido no padrão ISO.

A rotina de transporte deve rodar em paralelo as de tratamento ao usuário. Destarte, antes da biblioteca de transporte assim se tornar, deve disparar um processo totalmente assíncrono que chamará a rotina de transporte de verdade. A seguir o procedimento:

```
EXPORT APRESENTACAO-AO-TRANSPORTE
       , CONECTE
       , TRANSMITA
       , RECEBA
       , OUCA
       , DESCONECTE
       , TELEGRAMA
       , ESTADO-DA-CONEXAO
       , ALTERA-PARAMETROS
       , PROTOCOLO-DE-TRANSPORTE;

PROCEDURE TRANSPORTE; EXTERNAL;
TASK T-PROTOCOLO-DE-TRANSPORTE;

IF  T-PROTOCOLO-DE-TRANSPORTE.STATUS < 0
THEN BEGIN        % foi disparado por usuário e não pelo operador
     REPLACE T-PROTOCOLO-DE-TRANSPORTE.NAME
         BY "OBJECT/SUP/TRANSPORTE/TASK.";
     RUN TRANSPORTE [T-PROTOCOLO-DE-TRANSPORTE];
```

```
                % disparou assincronamente a rotina
                END

FREEZE (PERMANENT);        % AGORA SOU UMA BIBLIOTECA
```

FIGURA 6.6 Disparo da rotina de transporte

Assim o sistema de transporte será iniciado por um programa chamando uma das rotinas de transporte ou, numa instalação mais precavida, com o comando

```
RUN OBJECT/SUP/TRANSPORTE/TASK
```

executado pelo operador.

Tal processo é muito simples pois nada mais é que uma chamada à rotina de transporte.

```
BEGIN
LIBRARY TRANSPORTE (TITLE= "OBJECT/SUP/TRANSPORTE/LIB.");
PROCEDURE PROTOCOLO-DE-TRANSPORTE (INFO);
   STRING INFO;
   LIBRARY TRANSPORTE;
STRING MSG;      % para implementações futuras
PROTOCOLO-DE-TRANSPORTE(MSG);    % ou chamada
END OF PROGRAM
```

FIGURA 6.7 O processo de transporte

Devido a necessidades de testes a simplicidade não é o que parece. O programa verdadeiro será visto adiante.

6.5.1 Bloco externo

Como linguagens do tipo Algol são estruturadas em blocos o corpo do programa será o bloco mais externo dos escritos.

Neste caso é um laço de duração infinita (ou até receber uma mensagem do operador solicitando o encerramento do transporte) em que é disparado o procedimento de leitura assíncrona do nível de rede e uma espera pela ocorrência de um evento determinante da atitude a seguir:

1) FATIA-MÍNIMA - uma fatia de tempo que, se neste período nada ocorreu, indica que um temporizador estourou seu prazo, visto esta fatia-mínima ser calculada como hora do primeiro temporizador a estourar menos hora atual;

2) INTERFACE.CHANGEEVENT - troca do estado de algum sub-arquivo, pode ser encerramento de uma conexão ou abertura de uma nova;

3) INTERVACE.INPUTEVENT - algum usuário enviou uma UDST, logo descobre qual a obtém e a gerencia;

4) ALGO-DA-REDE - uma UDSN, após verificar se a UDSN destina-se ao transporte, trata a UDSN;

5) CANAL-EXPRESSO.QINSERTEVENT - a UDST veio através da fila de dados expressos, logo deve ser tratada pelo mesmo conjunto de rotinas que tratam da recebida através do interface;

6) MYSELF.EXCEPTIONEVENT - o operador do sistema utilizou o comando HI para dar instruções ao protocolo; usado para depuração e informações de gerenciamento.

Após procura um temporizador em que tenha ocorrido TIME-OUT para, só aí, retornar ao início do laço.

6.5.2 Gerenciamento do interface do usuário

Recebe a mensagem do usuário identificando a conexão, o evento indicado e a partir do estado atual da conexão, toma um dos seguintes procedimentos:

a) EVENTO=TCONREQ
ESTADO=CLOSED

Se não houver uma conexão de rede disponível ainda, então a requisite.

Caso houver transmita CR e guarde nome do processo par desejado, e tome ação 4(ações descritas na seção 6.3.6).

Próximo estado = WFCC.

b) EVENTO=TCONRESP
ESTADO=WFTRESP
Transmita CC e tome ação 4
Próximo estado = AKWAIT.

c) EVENTO=TDISREQ

ESTADO=WFNC
Próximo estado = WFCC.

ESTADO=WETRESP, AKWAIT, OPEN
Próximo estado = CLOSING e tome ação 4.

ESTADO=WFCC
Próximo estado = WBCL, e tome ação 3.

d) EVENTO=TOTREQ

ESTADO=AKWAIT
Construa e armazene UDPT.

ESTADO=SCLOSEDW, WFEA
Armazene UDPT.

ESTADO=OPEN
Construa e armazene UDPT.
Transmita UDPTs armazenadas até o limite da janela.

Se a janela estiver fechada então próximo es-

tado = SCLOSEDW.

e) EVENTO=TEXREQ

ESTADO=OPEN
Transmita ED.

ESTADO=SCLOSEDW
Tome ações 4 e 3.
Próximo estado = WFEA.

ESTADO=WFEA
Armazene dados expressos.

6.5.3 Tratamento de um evento de temporização

Sempre que ocorrer um time-out é necessário a execução de uma rotina que segue o diagrama de estado proposto pelo padrão ISO. O temporizador é imediatamente tornado inativo após o estouro.

a) EVENTO= T-REF

Referências congeladas, usado apenas no estado REFWAIT, visa esperar que decorra um certo período até que UDPTs mais atrasadas possam chegar com tal período, há a segurança de que as mensagens da atual conexão não irão interferir na que vier ocupar o lugar desta. O próximo estado será CLOSED e ativar-se-ão os procedimentos de encerramento de conexão.

b) EVENTO = T-RETRANS

Proteção contra perda de UDPTs não sinalizada.

ESTADO = WBCL, WFCC

Se o número máximo permitido de retransmissões não foi atingido, então transmita CR e tome ação 4, senão e o

estado for WFCC, transmita DR, tome ação 4, informe ao usuário uma desconexão (TDISIND) e próximo estado = CLOSING.

ESTADO = AKWAIT

Se o número máximo permitido de retransmissões ainda não foi atingido, retransmita CC e tome ação 4, senão desista, isto é, transmita DR, informe a desconexão(TDISIND), tome ação 4 e próximo estado = CLOSING.

ESTADO = CLOSING

Se o número permitido de retransmissões ultrapassou o máximo permitido, então transmita DR e tome ação 4, senão tome ação 1, e próximo estado = REFWAIT.

ESTADO = WFCC-R, OPEN-R

Novamente, se ainda é possível retransmitir, requisite uma nova conexão de rede (NCONREQ) e tome ação 4,se não requisite desconexão de rede (NDISREQ) e informe ao usuário a desconexão (TDISIND), tomando, por fim, ação 1.

ESTADO = OPEN

Se houver uma UDPT ED armazenada, transmita-a, se não transmita toda a janela aguardando reconhecimento.

c) EVENTO = T-INACT

Inatividade, se o intervalo de inatividade expirar deve ser feita a liberação da conexão. Para precaver-se durante longos períodos em que a inatividade é devida a uma ausência normal de dados a entidade de transporte utiliza-se do temporizador de janela.

ESTADO = AKWAIT, WBCL

Tome ação 1. Próximo estado = REFWAIT.

ESTADO = WFTRESP

Informe ao usuário a desconexão (TDISIND), tome ação 1 e próximo estado = REFWAIT.

ESTADO = OPEN

Informe ao usuário a desconexão (TDISIND), tome ação 4, transmita DR e próximo estado = CLOSING.

d) EVENTO = T-WINDOW

Temporizador de janela, compõe o intervalo máximo entre atualizações de janela. Isto é feito para que a entidade remota de transporte não pressinta erroneamente falta de atividade e desconecte-se.

ESTADO = CLOSING, REFWAIT, OPEN, WFEA, SCLOSEDW

Transmita AK e tome ação 2.

A ação para os dois primeiros estados não está definida no padrão ISO. Outra solução seria colocar este temporizador como inativo.

6.5.4 Tratamento de uma UDPT que chegou

Após a validação de cada UDPT componente da UDSR, é realizado um tratamento especial sobre cada uma isoladamente, a partir da obtenção do índice da conexão a qual ela é dirigida o processo é realizado segundo um cruzamento Estado versus Evento.

a) ESTADO = CLOSED

Só não ignora um evento CR que, após sua verificação e procedimentos para atualização dos registros da conexão, deve informar ao usuário uma indicação de conexão

(TCONIND) e fazer próximo estado = WFTRESP.

b) ESTADO = WBCL

Espera-se CC para desconectar pois deve ter ocorrido um TDISREQ enquanto o esperava.

EVENTO = CC

Tome ação 4, transmita DR e próximo estado= CLOSING.

EVENTO = CR, ER

Tome ação 1.

c) ESTADO = WFCC

EVENTO = CC

Se os parâmetros da UDPT CC forem aceitáveis,transmita AK, informe a confirmação da conexão, zere o contador de retransmissões, assim como o seu temporizador e tome ações 2 e 3, próximo estado = OPEN.

Senão, informe ao usuário a desconexão (TDISIND), tome ação 1 e transmita DR, próximo estado = CLOSING.

EVENTO = DR

Informe ao usuário a desconexão, tome ação 1 e próximo estado = REFWAIT.

EVENTO = ER

Tome ação 1, próximo estado = REFWAIT.

d) ESTADO = WFTRESP

EVENTO = CR

Tome ação 9, isto é, nada faça pois deve ser uma retransmissão simples.

EVENTO = DR

Informe ao usuário a desistência do processo par e conseqüente desconexão (TDISIND), transmita DC e tome ação 1.

e) ESTADO = AKWAIT

Após a entidade de transporte ter aceito a conexão e enviado um CC, deve ficar aguardando um AK para o início da fase de transferência de dados - é a confirmação pela entidade par da recepção do CC.

EVENTO = AK

Inicie todos os temporizadores, transmita UDPTs armazenadas, se houver, até o limite permitido pela janela, próximo estado = OPEN.

EVENTO = DT

Se a seqüência da UDPT não estiver dentro da janela de recepção, ignore-a; caso contrário, tome ações 2 e 3, se houver UDPT ED armazenada transmita AK e, após, transmita-a, próximo estado = WFEA, senão, transmita AK e próximo estado = OPEN.

EVENTO = ED

Informe ao usuário a existência de dados expressos, tome ações 2 e 3; se houver dado expresso para ser transmitido, faça-o e próximo estado = WFEA, senão, transmi

ta EA e próximo estado = OPEN.

EVENTO = CR

Transmita CC (provavelmente não chegou o CC anterior e o CR estará sendo retransmitido). Próximo estado = AKWAIT.

EVENTO = ER

Informe a desconexão (TDISIND), tome ação 4, próximo estado = CLOSING.

f) ESTADO = CLOSING

Deve ser aguardado o estouro de retransmissões, logo todas as UDPTs serão ignoradas.

g) ESTADO = REFWAIT

Como no estado anterior, é um temporizador que indicará a saída deste estado, exceto se vir um DR que deverá ser respondido com um DR.

h) ESTADO = OPEN

EVENTO = CR, EA

Tome ação 3, não há definição no padrão para o caso EA, julgamos aqui a melhor técnica.

EVENTO = AK

Faça crédito da conexão = campo CDT da UDPT.

Libere as UDPTs anteriores ao AK.

Incremente a seqüência máxima de transmissão e tome ações 5 e 3.

Se a janela estiver fechada então próximo estado= SCLOSEDW, senão transmita UDPTs armazenadas até o limite da janela.

i) ESTADO = WFEA

EVENTO = EA

Se houver dados expressos a transmitir, libere UDPT ED, transmita-a e tome ações 4 e 3, caso contrário, próximo estado = OPEN se janela não estiver fechada, senão SCLOSEDW.

EVENTO = AK

Libere UDPTs, tome ação 6.

j) ESTADO = SCLOSEDW

EVENTO = AK

Libere UDPTs, se ainda é possível transmitir mais UDPTs sem ferir o limite imposto pela janela então transmita UDPTs armazenadas até, no máximo, este limite e faça próximo estado = OPEN.

EVENTO = EA

Libere a UDPT ED transmitida.

k) ESTADO = OPEN, WFEA, SCLOSEDW

EVENTO = DT

Se o bit EDT estiver ligado, então é necessário montar as UDPTs de tal forma que seja possível recompor a mensagem original: INDICE-DADO-FINAL = T-NR, INDICE-PRÓXIMO-DADO-INICIAL = T-NR+1.

Insira a mensagem recebida na lista de mensagens

recebidas.

Tome ações 5, 3 e 2 e se toda a mensagem já estiver "preenchida", informe ao usuário (TDISIND).

Transmita AK.

6.5.5 Diagrama de estados

Conforme foi possível reconhecer, a implementação procurou respeitar ao máximo o diagrama de estados sugerido pela ISO e, de fato, tal objetivo foi bem sucedido.

Portanto, esse diagrama será o mesmo da seção 3.9 e descrito de uma forma mais detalhada neste capítulo.

7 TESTES, DESEMPENHO E CONSIDERAÇÕES FINAIS

Este capítulo trata da avaliação e desempenho do produto final gerado. Esta avaliação corresponde à validação e testes de qualidade, assim como o instrumental necessário para aplicá-los. Após uma análise do desempenho pretende orientar um futuro usuário dos recursos que o mesmo demanda e em que condições deve ser aplicado. Por fim, são realizadas sugestões que possam aprimorar sua qualidade ou ajustá-lo às eventuais necessidades /CAB 85/ e /ISO 85/.

7.1 O processo de validação

Se a validação de um protocolo já é uma tarefa extremamente complexa, envolvendo inúmeras teorias e trabalhos, a validação da implementação deste protocolo sofre em grau maior de tais insuficiências.

Não se pretendeu, portanto, validar o protocolo de transporte proposto pela ISO - o que seria muita pretensão - já "semi-validado" pelo ICST (Institute of Computer Science and Technology) e, sim, testar e validar a implementação do protocolo tal como ela foi proposta e, diante dos impasses surgidos no protocolo ISO, se houver, tomar as atitudes cabíveis.

O ICST possui um produto capaz de testar implementações deste protocolo mas, infelizmente, não nos foi possível utilizá-lo, por razões óbvias, no entanto no V Workshop sobre protocolos de transporte para redes locais, em 1984, é definida uma bateria de testes /NAT 84/.

Além de tais, o processo permaneceu em execução durante seis meses com simulações desde falhas físicas até solicitações erradas de usuários ou abraços mortais em solicitações.

Neste processo, transferiram arquivos de texto e código objeto, mensagens inter-processos e até fitas intei-

ras, foram testadas conexões múltiplas e com diversos graus de ocupação de ambos equipamentos. Permitiu-se até a conexão a si próprio como uma alternativa viável.

Em tais comunicações foram utilizados diversos monitores de teleprocessamento (MCS) - GEMCOS - transacional, CANDE - editor de programas e uso geral e RJE entrada remota de JOBs não sendo aprovado este último. Foi necessário também escrever um protocolo de enlace visto as instalações Renner (onde realizaram-se os testes) não possuírem protocolos computador a computador, tampouco com redes locais.

Naturalmente, para os testes iniciais e, mesmo por fim, quando o processo era muito pesado ou para não depender de dois equipamentos, optou-se por utilizar apenas e um e simular o nível de rede. Este nível estaria dentro da tarefa que chama a rotina de transporte e seria disparado por esta. Mais tarde tal poderia ser utilizada no AE pois este possui um conceito de subsistema (para aumento de memória) no qual programas contidos num não podem ver programas de outro - o transporte resolve tal. Com isto aquela forma de visualização global do transporte fica modificada.

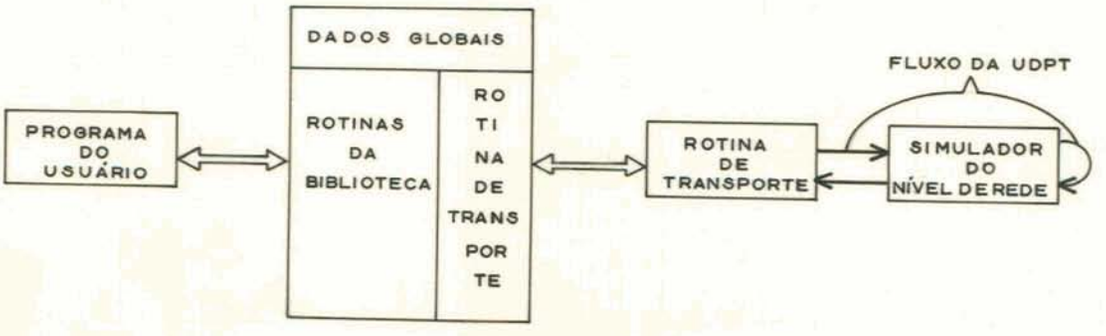


FIGURA 7.1 Visão geral do protocolo de transporte com simulação de rede

E, também o processo de transporte sofre alterações passando a ficar dessa forma:

```
BEGIN

 LIBRARY TRANSPORTE(TITLE="OBJECT/SUP/TRANSPORTE/LIB.");

 PROCEDURE PROTOCOLO_DE_TRANSPORTE(INFO);
           STRING INFO;
           LIBRARY TRANSPORTE;
 STRING MSG;
 TASK T;

 $ SET OMIT = NOT SIMULANIVELDEREDE
 PROCEDURE NIVELDEREDE;    % uma simulação do nível de  rede
    BEGIN                  % onde a mensagem é transmitida pa-
    REAL CONT,AUX,         % ra o próprio equipamento transmissor
    FILE REDE(KIND=PORT,                 % será com intervalo
              BLOCKSTRUCTURE=EXTERNAL,   % tipo porta
              FRAMESIZE=8,               % estamos evitando,
              MAXSUBFILES=i,             % assim, um  consumo
              MYUSE=IO,                  % exagerado de recur
              TITLE="LOCALNETLEVEL3A.",  % sos para um   sim-
              MYNAME="NIVEL3.",          % ples teste
              YOURNAME="NIVEL4.");
    ARRAY MSGEØ:259];
    FILE D(KIND=DISK,MAXRECSIZE=36Ø,NESWILE, % arquivo   em
           FLEXIBLE=TRUE,                    % disco   para
           PROTECTION=SAVE,                  % depuração
           TITLE="DTST/TRANSPORTE.");
    EPILOG PROCEDURE P;    % o arquivo deve ser sempre fecha
           BEGIN           % do principalmente se o programa
           CLOSE(D,CRUNCH); % cair por erro
           DISPLAY("REDE REGS. : "CAT STRING(CONT,*));
           END;
    OPEN(REDE);
    WHILE TRUE DO
      BEGIN
      REPLACE MSG BY Ø FOR 26Ø WORDS;
```

```
      READ(REDE,1Ø24,MSG);          % obtém o registro envia-
      CONT:=*+1;                    % do à rede
      AUX:=REDE.CURRENTRECORD;
      WRITE(D,359,MSG);             % grava no arquivo de de-
      WRITE(REDE,AUX,MSG);          % puração
      END;
END NIVELDEREDE;
T.DECLAREDPRIORITY:=7Ø;
REPLACE T.NAME BY "SUP/NIVELDEREDE.";
PROCESS NIVELDEREDE[T];   % o emulador do nível de rede é  dis-
                            parado
PROTOCOLO-DE-TRANSPORTE(MSG);% rotina de transporte
DO WAITANDRESET(T.EXCEPTIONEVENT) UNTIL
                T.STATUS LEQ Ø;  % aguarda fim da rede
END OF PROGRAM.
```

FIGURA 7.2 O processo de transporte com simulação do nível de rede

O arquivo de nome D (na realidade DTST/TRANSPORTE) é um arquivo de depuração onde são gravadas todas as UDSRs geradas pelo protocolo.

Além disto, havia dentro da rotina de transporte ou mesmo nas rotinas do usuário procedimentos de depuração.

7.1.1 Interface da depuração com operador do sistema

Se dá através da instrução <número do processo> HI <valor>. <número do processo> é o número que a rotina de transporte recebeu do sistema operacional quando iniciou, <valor> é um número inteiro que indica o desejo do operador; pode ser:

Ø - liga o modo depuração;
1 - lista todos os temporizadores;
2 - mostra os registros de estado do TC corrente;
3 - mostra os nomes de todos os processos que "pe

garam" uma conexão;

4 - encerra o protocolo incondicionalmente;

5 - desliga o modo depuração.

Conforme implementação, tais resultados são colocados numa impressora.

7.1.2 Função Trace

A função Trace é uma rotina que mostra determinadas condições ao operador do sistema e, concomitante, na jornalização do processo.

Esta rotina deve ser tornada nula para uma implementação verdadeira (de produção).

7.1.3 Função Assert

É uma função que testa asserções nos laços e em pontos de entrada e saída de rotinas básicas. O cumprimento dessas premissas garante a integridade da programação.

É aconselhável que em ambiente que não o de desenvolvimento esta função seja esvaziada, na intenção de melhorar a performance.

7.1.4 Opção Statistics

Função executada pelo compilador, mostra a duração de cada rotina e quantas vezes esta foi executada, muito útil para análise de desempenho.

Como as demais deve estar desligada para executar em produção, sob pena de degradar o sistema.

7.2 Análise de desempenho

Este é um sistema implementado em equipamentos de grande porte e portanto os valores e as funções realizadas

participam de um percentual quase insignificante em relação ao total. Isto equivale a menos de 0,8% da memória do A9 e uma base de 1,5% do consumo de processador.

Como desempenho das funções que tendem a atender ao usuário, o desempenho também tem sido satisfatório.

É interessante fornecer alguns dados para uma aproximação do desempenho do sistema.

7.2.1 Premissas

a) Sistema: B6910 Advanced
0.7 MIPS
6.2 MB memória;

b) Ocupação de UPC: 98%;

c) Simulador de rede, isto é, enviando mensagens a si próprio para menor interação de fatores externos;

d) Opção Statistics, Trace e Debug ativadas;

e) 25 conexões permitidas;

f) 2 conexões de fato realmente abertas.

7.2.2 Resultados

Valores médios:

a) ocupação média de memória: 55 KBytes;

b) tempo de ida e chegada de uma UDST dados (128 octetos): 163 ms;

c) tempo de apresentação-ao-transporte: 560 ms;

d) tempo de abertura de conexão: 95,4 ms;

e) tempo de encerramento de conexão: 48,8 ms;

f) tempo de viagem de uma UDPT: 102 ms;

g) intervalo de residência na rotina de transporte: 30 ms;

h) tempo médio para varrer todos temporizadores: 0,92 ms;

i) transmissão de um ponte-linhas por minuto: 350.

7.3 Exemplo de um programa de teste

Este programa SUP/TRANSPORTE/T na realidade dispara P1 e P2 que comunicam-se entre si.

P1 fica aguardando que algum processo conecte - se a ele, a seguir transmita uma mensagem, espera uma, transmite a próxima, espera receber mais uma, transmite mais quatro e espera a recepção de uma final. Após, desconecta-se.

P2 procura conectar-se a LIB/P1, que é o nome de P1, e age simetricamente a P1. Antes de desconectar-se envia dado expresso.

```
BEGIN

LIBRARY LIB(TITLE="OBJECT/SUP/TRANSPORTE/LIB.");

REAL PROCEDURE APRESENTACAO_AO_TRANSPORTE(COMM);
        FILE COMM;
     LIBRARY LIB;

REAL PROCEDURE CONECTE(PORTA,ENDER,DESTINO,ESPERAR);
        VALUE ENDER,
              ESPERAR;
        FILE PORTA;
        REAL ENDER;
        STRING DESTINO;
```

```
        BOOLEAN ESPERAR;
     LIBRARY LIB;

REAL PROCEDURE TRANSMITA(PORTA,DADOS);
        FILE PORTA;
        STRING DADOS;

     LIBRARY LIB;

REAL PROCEDURE RECEBA(PORTA,DADOS,LIM);
        VALUE LIM;
        FILE PORTA;
        STRING DADOS;
        REAL LIM;
     LIBRARY LIB;

REAL PROCEDURE DESCONECTE(PORTA);
        FILE PORTA;
     LIBRARY LIB;

REAL PROCEDURE TELEGRAMA(PORTA,DADOS);
        FILE PORTA;
        STRING DADOS;
     LIBRARY LIB;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

PROCEDURE P1;    % um dos processos comunicantes
   BEGIN
   FILE F(KIND=PORT);
   REAL I,J,K,L;
   STRING S;
   I:=APRESENTACAO_AO_TRANSPORTE(F);

   J:=CONECTE(F,1,EMPTY,TRUE);
                %    %→aguarda que alguém conecte-se a ele
                % → endereço de rede
```

```
   FOR K:=1 STEP 1 UNTIL 8
   DO BEGIN
      L:=TRANSMITA(F,"TEXTO:" CAT STRING(K,1) CAT ":XMITIDO");
      IF K=2 OR K=4      % inverte abruptamente a direção das MSGS
      THEN BEGIN         % para maior diversidade do protocolo
            RECEBA(F,S,Ø); % espera ad aeternum se necessário, não é
            DISPLAY("MSG:" CAT S); % necessário utilizá-la     como
            END;                   % função embora recomendável
      END;

TASK T1,T2;
FILE LP (KIND = PRINTER);

   L:=RECEBA(S,S,Ø);     % uma última mensagem
   DIAPLAY(S);
   I:=DESCONECTE(F);
END  OF  P1;

PROCEDURE P2;         % o segundo processo
   BEGIN
   FILE F(KIND=PORT)  % o arquivo de comunicação com o TRANSPORTE
   REAL I,J,K,L;
   STRING S;
   STRING S1;
   I:=APRESENTACAO_AO_TRANSPORTE(F);

   J:=CONECTE(F,1,"(SUP)LIB/P1",TRUE);
                % → nome do processo desejado (P1)
   FOR K:=1 STEP 1 UNTIL 8
   DO BEGIN
      L:=RECEBA(F,S,Ø);
      DISPLAY("RECEBEU MSG DE "CAT STRING(L,*) CAT "BYTES:" CAT S);
      IF K="                % também inverte o fluxo, claro
      THEN TRANSMITA(F,"OK")
      ELSE
      IF K=4
      THEN BEGIN
```

```
            S1:="ESTAMOS REALIZANDO UMA TENTATIVA DE TRANS-"
                CAT "TIR UM TEXTO DE MAIS DE DUZENTOS     E"
                CAT "CINQUENTA E SEIS CARACTERES NO INTUITO"
                CAT "DE TESTAR DEFINITIVAMENTE TODOS      OS"
                CAT "REQUISITOS DA COMUNICACAO. ESTAMOS,POR"
                CAT "TANTO, TESTANDO AS ROTINAS DE  SEGMEN-"
                CAT "TACAO E CONCATENACAO SEPARANDO A  MEN-"
                CAT "SAGEM EM VARIOS PEDACOS E  ENVIANDO-AS"
                CAT "À ENTIDADE PAR QUE AO RECOLHER  DEVERÁ"
                CAT "CONCATENAR NOVAMENTE FORMANDO O  TEXTO"
                CAT "ORIGINAL. BOA SORTE.";
                DISPLAY("TRANSMITIU MSG DE "CAT STRING (LENGTH
                    "(S1),*) CAT CARACTERES");
            TRANSMITA(F,S1);
            END
    END;
    TELEGRAMA(F,"FINAL");      % um dado expresso
    I:=DESCONECTE(F);
END

 % colocação dos nomes em cada processo
REPLACE T1.NAME BY "LIB/P1.";
REPLACE T2.NAME BY "LIB/P2.";

 % disparo de cada processo
PROCESS P1[T1];
PROCESS P2[T2];

 % o corpo principal deve aguardar o término de seus dois
 % filhos
DO WAITANDRESET(MYSELF.EXCEPTIONEVENT)
   UNTIL T1.STATUS < Ø AND T2.STATUS < Ø;

END OF PROGRAM.
```

FIGURA 7.3 Sup/Transporte/T

7.4 Observações finais

Embora seja um sistema já implementado é de bom tom observar pequenos detalhes que possam contribuir para o desenvolvimento do produto.

Uma capacidade muito útil é a de conter uma biblioteca de programas que seriam disparados pelo próprio protocolo, realização que evitaria ter que disparar um processo em cada equipamento para realizar uma simples transferência de arquivos. Isto é tanto pior quando a função OUÇA não está implementada em sua plenitude.

A BURROUGHS estará lançando oficialmente, em breve, um monitor de teleprocessamento (MCS) que opera com a RENPAC (Rede Nacional de Pacotes) fazendo uso do protocolo X.25 para nível de rede. Embora tal fato não corresponda a uma rede local, nosso produto também se propõe a atendê-lo. Para tanto, será necessário, após o lançamento da Burroughs, um ajuste do seu interface com o nível de rede pois este X.25 não opera conforme os padrões para interface. Também será necessário adaptar o protocolo aos padrões definidos pela EMBRATEL quando esta o fizer. Pagou-se aqui o preço de adiantar-se aos órgãos oficiais.

Ademais, como o trabalho em conjunto a este não está completamente realizado, não foi possível efetuar testes utilizando a rede CETUS com microcomputadores ITAUTEC. Este foi o objetivo inicial do trabalho.

Por fim, cabe salientar que sendo este protocolo um padrão internacionalmente aceito e atendendo tanto redes locais como públicas, necessita um constante aperfeiçoamento e ajuste às características de uma área extremamente mutável.

8 CONCLUSÃO

Este trabalho compôs-se de um estudo do Padrão ISO para interconexão de sistemas abertos (OSI) e, a partir disto, a escolha de um Protocolo de Transporte que melhor se adaptasse às condições propostas a priori.

Para tal protocolo optou-se pelo próprio Padrão ISO, o que redundou num estudo detalhado deste, não só como objetivo da implantação bem como sua descrição com fins didáticos.

A sua implementação, primordialmente, visou atingir quatro metas:

a) provar sua validade, adequando-o nas condições exigidas pela Universidade;

b) obter uma implementação na qual futuros alunos ou até pesquisadores possam realizar estudos e experiências afins;

c) obter, através do nível de transporte, suporte para futuros trabalhos em níveis superiores tais como nível de sessão e aplicação;

d) utilizar e disseminar técnicas de implementação de protocolos de comunicação.

Durante a sua realização foram escritos e apresentados dois artigos técnicos, /MUS 85/ e /BRA 84/, em simpósio nacional e internacional, respectivamente, tendo como objetivo divulgar à comunidade acadêmico-científica a experiência adquirida na implementação de um modelo teórico criado como padrão. Em ambas as oportunidades houve receptiva aceitação do trabalho, com muitas perguntas sendo dirigidas ao apresentador com vistas à absorção maior dos conhecimentos adquiridos mediante o trabalho realizado.

Um fato importante a ressaltar e a sugestão aos mais interessados é a leitura do código fonte gerado neste trabalho, onde todas as decisões, estruturas e algoritmos estão documentados intrinsecamente.

Sugere-se, ainda, a leitura deste trabalho a todos que desejem estudar interconexões de equipamentos e, sobretudo, implementar níveis superiores.

ANEXO I

Rotina de interconexão entre B6910 e o A9

```
CHARACTER ALGORITHM INTERCONNECT BEGIN
   LINE CONTROL BEGIN
      CONTROL BLOCK CBIN, CBOUT;
      PROCESS LINE BEGIN
         INTEGER WAITRESULT:=Ø;
         WHILE TRUE DO BEGIN
            WHILE NOT LINE.ADAPTORREADY TO WAIT LINE.ADAPTOREVENT;
            IF LINE.HANGUP THEN BEGIN
               LINE.HANGUP:=FALSE;
               INITIATE(CBOUT,DISCONNECT);
               WAIT CBOUT;
            END; % IF
            WHILE NOT LINE.CONNECTED DO WAIT LINE.ADAPTOREVENT;
            WHILE NOT LINE.LNRDY DO WAIT LINE.SYSTEM;
            NEXTSTATION;
            WHILE NULLREFERENCE(STATIONREFERENCE) DO BEGIN
               WAIT LINE.SYSTEM;
               NEXTSTATION;
            END; % WHILE
            IF STATION.QUEUED COR NOT NULLREFERENCE(STATION.REQUEST)
               THEN BEGIN
               IF NULLREFERENCE(STATION.REQUEST) THEN NEXTREQUEST;
               MOVETEXT(REQUEST,CBOUT);
               INITIATE(CBOUT,OUTPUT);
               WAIT CBOUT;
               MOVETEXT(CBOUT,REQUEST);
               CASE CBOUT.CATEGORY BEGIN
                  COMPLETED : FINISHREQUEST;
                  ELSE :      ABORT BADCATEGORY
               END % CASE
            END ELSE
               IF STATION.ENABLED THEN BEGIN
                  ALLOCATE(CBIN,*);
                  INITIATE(CBIN,INPUT);
```

```
            CASE WAIT(*,CBIN,LINE.SYSTEM) BEGIN
               1: SKIP;
               2: BEGIN
                  INITIATE(CBOUT,CANCEL INPUT);
                  WAIT CBOUT;
                  WAIT CBIN;
                  END
            END; % CASE
            CASE CBOUT.CATEGORY BEGIN
               CANCELLED: DEALLOCATE CBIN;
               COMPLETED: BEGIN
                  NEWRESULT;
                  MOVETEXT(CBIN,RESULT);
                  SENDHOST INPUT;
                  END;
               ELSE: ABORT BADCATEGORY;
            END  % CASE
         END  % ELSE BEGIN
      END  % WHILE LOOP
   END ENDPROCESS LINE
END; % LINE CONTROL
ADAPTOR CONTROL BEGIN
   PROCESS INPUT BEGIN
      INTEGER NCR:=Ø;
      IF RECEIVER.ENABLED THEN DISABLE RECEIVER;
      ENABLE RECEIVER;
      RECEIVE(*) NCR;
      NOCANCEL:=TRUE;
      IF NCR NEQ 4'FFF' THEN BEGIN
         STORE NCR;
         RECEIVE(*) TEXT UNTIL BREAK;
      END;
      DISABLE RECEIVER;
   END ENDPROCESS INPUT;
   PROCESS OUTPUT BEGIN
      IF RECEIVER.ENABLED THEN DISABLE RECEIVER;
      ENABLE TRANSMITTER;
```

```
            TRANSMIT TEXT UNTIL BREAK;
            DISABLE TRANSMITTER;
        END ENDPROCESS OUTPUT
    END  % ADAPTOR CONTROL
  END; % ALGORITHM
END MODULE; % END OF PROTOCOL MODULE
```

ANEXO 2

Rotina para nível de enlace com rede CETUS-B6700

```
CONTROL INT CETUS:
   10:  IF LINE(READY)
        THEN IF STATION(VALID)
        THEN IF STATION(ENABLE)
        THEN IF STATION(READY)
             THEN IF STATION(QUEUED)
                  THEN INITIATE REQUEST
                  ELSE INITIATE ENABLEINPUT.
        PAUSE.
        GO TO 1Ø.

REQUEST CETUS B6700 TRANSMIT:

        INITIATE TRANSMIT.

   10:  FETCH[20].
        TRANSMIT CHARACTER [BREAK:30]
        GO TO 1Ø.
   20:  FINISH TRANSMIT.
        TERMINATE NORMAL
   30:  TERMINATE ENABLEINPUT.

REQUEST CETUS B6700 RECEIVE:
   ERROR [Ø] = BUFOVFL: 30
               LOSSOFCARRIER: 20
               PARITY: ABORT
               STOPBIT: 30
               TIMEOUT: 20.
        INITIATE RECEIVE.
   10:  RECEIVE (150MILLI) ERROR [Ø].
        STORE [GETSPACE:30]
        GO TO 1Ø.
```

```
20:  FINISH RECEIVE
     TERMINATE NORMAL.
30:  TERMINATE ERROR.
```

BIBLIOGRAFIA

/AND 84/ ANDREONI, Gaetano. Fortran interface to X.25 and transport service. Computer Networks, London, 8(1):17-22, Fev. 1984.

/BRA 84/ BRANCO, Ricardo. Redes locais, exigências a um soft ware de apoio. In: CONFERÊNCIA LATINO-AMERICANA DE INFORMÁTICA, 10, Valparaíso, Abril, 26-29, 1984. Anales. Valparaíso, CLEI, 1984. p.45-50.

/BUR 84/ BURG, Fred et alii. Of local networks, protocols and the OSI reference model. Data Communications, New York, 13(13):129-50, Nov. 1984.

/BUR 77/ BURROUGHS CORPORATION. NDL Reference Manual. Set. 1977.

/BUR 80/ _____. DCALGOL Reference Manual. Set. 1980.

/BUR 81/ _____. ALGOL Reference Manual. Set. 1981.

/BUR 82/ _____. NDL II Reference Manual. Mar. 1982.

/BUR 83/ _____. DNOTES - Implementation Features relative to Mark 3.3.520. Jul. 1983.

/BUR 84/ _____. A9 System Reference Manual. Abr. 1984.

/BUR 84a/ _____. DNOTES - Implementation Features relative to Mark 3.4.750. Jul. 1984.

/BUR 84b/ _____. Cobol a Series Reference Manual. Dez. 1984.

/BUR 84c/ _____. DCALGOL a Series Reference Manual. Dez. 1984.

/BUR 84d/ _____. I/O Subsystem a Series Reference Manual. Dez. 1984.

/BUR 85/ _____. DNOTES - Implementation Features relative to Mark 3.5.200. Mai. 1985.

/CAB 85/ CABRAL, M.I. & SAUVÉ, J. Modelagem de múltiplas conexões de transporte com controle de fluxo de janela. In: SIMPÓSIO BRASILEIRO SOBRE REDES DE COMPUTADORES, 3, Rio de Janeiro, Abril 1-3, 1985. Anais. Rio de Janeiro, SBC/UFRJ/LARC, 1985. p.16.1-16.14.

/CRU 84/ CRUZ, Fernando. Estudo comparativo entre propostas de protocolo de transporte. Porto Alegre, CPGCC UFRGS, jan. 1984.

/CUN 85/ CUNHA, P. et alii. Aspectos da implementação de um protocolo de transporte num ambiente seqüencial. In: SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 12. Porto Alegre, Julho 20-27, 1985. Anais. Porto Alegre, SBC/CLEI/UFRGS, 1985. p. 1-10.

/CUN 85a/ _____. Implementação de um protocolo de transporte para a rede CEPINNE. In: SIMPÓSIO BRASILEIRO SOBRE REDES DE COMPUTADORES, 3. Rio de Janeiro, Abril 1-3, 1985. Anais. Rio de Janeiro, SBC/UFRJ/LARC, 1985. p.9.1-9.26.

/DEI 84/ DEITEL, Harvey. An Introduction operating systems. Massachusetts, Addison-Wesley, 1984.

/FER 85/ FERREIRA, Manoelito. Padronização de protocolos de alto nível. In: SIMPÓSIO INTERNACIONAL DE TELEMÁTICA. Porto Alegre, Maio 27-30, 1985. Anais. Porto Alegre, UFRGS/IFIP, 1985.

/FUJ 84/ FUJII, E. et alii. MULTIPLUS - Sistema operacional para uma rede local. Boletim Scopus, São Paulo, 6(69):1-8, Mar. 1984.

/GIE 78/ GIEN, Michel. A File transfer protocol (FTP). Computer Networks, London, 2(4):312-19, Set. 1978.

/INT 80/ INTERNATIONAL ORGANIZATION FOR STANDARTIZATION. Data processing - open systems interconnection - basic reference model. ISO/TC 97/SC 16 DP 7498, New York, Dez. 1980.

/INT 83/ _____. Transport service definition. ISO/TC 97/SC 16 DIS 8072, Ottawa, Out. 1983.

/INT 84/ _____. Connection oriented transport protocol specification. ISO/TC 97/SC 16 DIS 8073, Jan. 1984.

/INT 83a/ _____. Protocol for providing the connectionless network service. ISO/TC 97/SC 6 DP 8473, Tianjin, Set. 1983.

/INS 82/ INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. Local area network standard - logical link control. Nov. 1982.

/INS 82a/ _____. Local area network standard - CSMA/CD access method and physical layer specifications. Dez. 1982.

/ISO 85/ ISONI, M. et alii. Uma experiência em implementação e teste de protocolo para redes de computadores. In: SIMPÓSIO BRASILEIRO SOBRE REDES DE COMPUTADORES, 3. Rio de Janeiro. Abril 1-3, 1985. Anais. Rio de Janeiro, SBC/UFRG/LARC, 1985. p.12.1-12.6.

/ITA 85/ ITAÚ TECNOLOGIA. Rede local Itautec. Nov.1985.

/LAN 84/ LANGSFORD, Alwyn. The Open system user's programming interfaces. Computer Networks, London, 8 (1):3-12, Fev. 1984.

/LEI 85/ LEITE, J.R. & JINO, M. Uma estação de transporte de dados. In: SIMPÓSIO BRASILEIRO SOBRE REDES DE COMPUTADORES, 3. Rio de Janeiro, Abrill-3, 1985. Anais. Rio de Janeiro, SBC/UFRJ/LARC, 1985. p.7.1-7.14.

/LIN 84/ LININGTON, Peter. The virtual filestore concept. Computer Networks, London, 8(1): 13-6, Fev. 1984.

/MED 85/ MEDEIROS, J. & MENASCÉ, D. SNPUC - Um servidor de nomes para o REDPUC. In: SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 12. Porto Alegre, Julho 20-27, 1985. Anais. Porto Alegre, SBC/CLEI/UFRGS, 1985. p.11-29.

/MON 84/ MONTEIRO, José. Uma Técnica para o desenvolvimento e implementação de protocolos. In: SIMPÓSIO BRASILEIRO SOBRE REDES DE COMPUTADORES,2. Campina Grande, Abril 16-8, 1984. Anais. Campina Grande, SBC/UFPB, 1985. p.7.1-7.21.

/MUS 85/ MUSSE, J. et alii. Uma Análise do serviço de transporte. In: SIMPÓSIO BRASILEIRO SOBRE REDES DE COMPUTADORES, 3. Rio de Janeiro, Abril 1-3, 1985. Anais. Rio de Janeiro, SBC/UFRJ/LARC, 1985. p.17.1-17.14.

/NAV 85/ NAVARRO, L.C. & RIBEIRO, H. Protocolos de comunicação da rede local SCOPUS. In: SIMPÓSIO BRASILEIRO SOBRE REDES DE COMPUTADORES, 3.Rio de Janeiro, Abril 1-3, 1985. Anais. Rio de Janeiro, SBC/UFRJ/LARC, 1985. p.5.1-5.41.

/PAL 84/ PALLAZO, S. & ANDREONI, G. The programmatic interface - access to OSI. Computer Communications, London, 7 (2):79-85, Abr. 1984.

/RIB 84/ RIBEIRO, F.H. & NAVARRO, L.C. Protocolo de comunicação da Rede Scopus. Boletim Scopus, São Paulo, 6(69):9-20, Mar. 1984.

/SOU 85/ SOUZA, Wanderley. Utilização dos conceitos de módulo, porta e canal em especificações formais de serviços, protocolos e interfaces de comunicação. In: SIMPÓSIO BRASILEIRO SOBRE REDES DE COMPUTADORES. Rio de Janeiro, Abril 1-3,1985. Anais. Rio de Janeiro, SBC/UFRJ/LARC, 1985. p.25.1-25.23.

/TAN 81/ TANENBAUM, Andrew. Computer Networks. Englewood Cliffs, Prentice-Hall, 1981.

/TAR 83/ TAROUCO, Liane. Projeto de redes de teleprocessamento. Porto Alegre, PGCC da UFRGS, 1983.

/TAR 77/ _____. Um Estudo sobre redes de comunicação de dados. Porto Alegre, CPGCC UFRGS, 1977.

/TAV 85/ TAVARES, O. & SCHWARE, D. Uma Implantação de um protocolo de transferência de arquivos para redes de computadores. In: SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 12. Porto Alegre, Abril 1-3, 1985. Anais. Porto Alegre, SBC/CLEI/UFRGS, 1985. p.41-50.

/WEB 85/ WEBER, Kival. A Padronização de protocolos como um dos instrumentos da política nacional de informática. In: SIMPÓSIO BRASILEIRO SOBRE REDES DE COMPUTADORES, 3. Rio de Janeiro, Abril 1-3, 1985. Anais. Rio de Janeiro, SBC/UFRJ/LARC, 1985. p.II.1-II.9.

/WOR 84/ WORKSHOP FOR LOCAL AREA NETWORK - TRANSPORT, Washington, Feb. 1983 - Mar.1984 - Proceedings. Washington, U.S. Dept. Of Commerce/National Bureau Of Standards, 1984.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Pós-Graduação em Ciência da Computação

Protocolo de Transporte para
Apoiar a Interconexão com
Computadores de Grande Porte

Dissertação apresentada aos Srs.

Liane M R Tarouco

Profª Liane Margarida R. Tarouco

Lucas Moscato

Prof. Dr. Lucas Antonio Moscato

Prof. Dr. José Palazzo M. de Oliveira

Visto e permitida a impressão
Porto Alegre, .08./.10./.86.

Robert A. Price

Prof. Dr. Roberto Tom Price
Coordenador CPGCC