3º SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES (3º SBRC)

GRUPO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUIDOS DA

UNIVERSIDADE FEDERAL DE PERNAMBUCO

REDIS/UFPE

# IMPLEMENTACAO DE UM PROTOCOLO DE TRANSPORTE PARA A REDE CEPINNE

RECIFE

DEZEMBRO 1984

# IMPLEMENTACAO DE UM PROTOCOLO DE TRANSPORTE PARA A REDE CEPINNE

Paulo Roberto Freire Cunha (*)

José Augusto Suruagy Monteiro (**)

Emilio de Barros Lucena (***)

RESUMO

O presente trabalho tem o objetivo de descrever a experiência de implementação de um protocolo de Transporte para a Rede CEPINNE em um sistema DEC-10. Serão descritos basicamente o ambiente computacional disponível e a opção de implementação adotada face a este ambiente.

(*) Professor Adjunto do Departamento de Informática da UFPE.
(**) Professor Assistente do Departamento de Informática da UFPE.
(***) Aluno de Mestrado do Departamento de Informática da UFPE.

## 1. INTRODUCAO

Em Janeiro de 1981 a SEI (Secretaria Especial de Informática) elaborou um projeto [7] que objetivava a criação de um ambiente propício para o desenvolvimento da Teleinformática nas diversas universidades federais do Norte-Nordeste: Projeto CEPINNE (Centro Piloto de Serviços Públicos de Teleinformática para aplicações em Ciência e Tecnologia - Região Norte-Nordeste).

Dentre os objetivos específicos do projeto CEPINNE, encontravam-se o estabelecimento de modelo institucional para exploração de serviços de Teleinformática; a instalação e uso de minicomputadores nacionais em rede nas universidades; a otimização dos recursos de informática das universidades; a integração à Rede Pública de Pacotes; o desenvolvimento de Recursos Humanos especializados; os testes dos protocolos padronizados de comunicação de dados; e outros.

A Rede CEPINNE, que interligaria os CPDs das diversas universidades federais do Norte-Nordeste, seria composta de quatro nós de comutação, situados em Belém, Fortaleza, Recife e Salvador. Os CPDs das demais universidades seriam conectados à Rede através de concentradores regionais.

Por razões diversas o Projeto CEPINNE não obteve o desenvolvimento acelerado que se esperava e a Rede CEPINNE não pode ser implantada segundo o cronograma previsto. Em Janeiro de 1984, foi celebrado um convênio entre a EMBRATEL, a Universidade Federal de Pernambuco, a Universidade Federal da Paraíba e a Fundação de

Apoio a Pesquisa e Extensão da Universidade Federal da Paraíba (FUNAPE), cujo objetivo principal era o desenvolvimento e implementação de um protocolo de Transporte que ajudaria a viabilizar a existência da Rede CEPINNE. O convênio teria a EMBRATEL como órgão financiador e caberia à FUNAPE coordenar e repassar os recursos alocados.

O Projeto CEPINNE original foi revisto e ficou estabelecido que seriam instalados, inicialmente, apenas dois nós de comutação, sendo um em Belém e outro em Recife. O convênio firmado com as universidades UFPE e UFPB determinava que ambas deveriam implementar, nas suas máquinas hospedeiras, uma Estação de Transporte, tendo como referência a especificação de um Protocolo de Transporte desenvolvido pela PUC/RJ [4]. Caberia à EMBRATEL fornecer às universidades envolvidas conversores X.25 para permitir a conexão entre os computadores hospedeiros e a subrede de comunicação, e instalar nós REXPAC a fim de possibilitar a efetuação dos testes integrados do Protocolo de Transporte implementado.

Este artigo visa retratar a experiência de implementação da Estação de Transporte para a Rede acima citada, no sistema computacional disponível na UFPE. Na seção 2 faremos uma breve descrição acerca do Nível de Transporte, situando-o na arquitetura do Modelo de Referência da ISO. A seção 3 tem o objetivo de descrever rapidamente o protocolo de Transporte escolhido e os serviços que seriam implementados nesta versão preliminar. A seção 4 consiste de uma descrição pormenorizada da nossa implementação em

particular. Finalmente, na seção 5 faremos uma análise crítica da experiência e identificaremos direções para futuros trabalhos em relação a este projeto.

## 2. O NIVEL DE TRANSPORTE

A principal tarefa do nível de Transporte, que corresponde ao Nível 4 na arquitetura do Modelo de Referência para Interconexão de Sistemas Abertos da ISO ('International Organization for Standardization') [9], é prover uma transferência de dados fim a fim transparente, confiável e eficiente [8]. Tal transferência envolve o fornecimento de serviços tais como multiplexação de circuitos virtuais, quebra e remontagem de mensagens, controle de fluxo, controle de erros fim a fim, recuperação de falhas (RESET's e RESTART's), estabelecimento e encerramento de conexões, interconexão de redes, entre outros.

O fator de maior importância no projeto de um Protocolo de Transporte é o tipo de serviço fornecido pela subrede de comunicação [8]. Se a subrede provê um serviço de circuito virtual perfeito, que nos garante a entrega dos dados na ordem, sem perda ou duplicação, então nosso Protocolo de Transporte se torna bastante simples. Entretanto, dizer que uma dada subrede é confiável, pode significar dizer que a freqüência de ocorrência de RESET's, por exemplo, é da ordem de um em cada mil anos. Conseqüentemente, torna-se necessária a inserção de serviços complementares no nosso Protocolo de Transporte, haja vista que a

recuperação, quando da ocorrência de RESET's ou RESTART's, deverá ser feita pelas camadas superiores.

## 3. O PROTOCOLO DE TRANSPORTE PARA A REDE CEPINNE

Como Já dissemos anteriormente, ficou estabelecido que o protocolo de Transporte a ser implementado para a Rede CEPINNE deveria ser baseado, em linhas gerais, naquele especificado pela PUC/RJ [4], a fim de permitir uma aceleração no desenvolvimento do projeto.

De um modo geral, os serviços que este protocolo de transporte se propõe a prover podem ser subdivididos em dois grupos. O primeiro, Gerenciamento de Conexões, envolve os serviços de estabelecimento de conexões, encerramento (unilateral) de conexões, aborto de conexões e estado de conexões. O segundo, Transferência de Informações, envolve os serviços de transferência normal de dados, transferência de dados expressos (INTERRUPÇÕES) e transferência de Telegramas. Analogamente, tal protocolo necessita do fornecimento, pelo nível de Rede, de basicamente dois grupos de serviços. Gerenciamento de Circuitos Virtuais e transferência de dados. O gerenciamento de circuitos virtuais envolve serviços tais como estabelecimento e encerramento de circuitos virtuais, limpeza (RESET) de circuitos virtuais e serviço de RESTART.

O Protocolo de Transporte em questão é baseado na troca de fragmentos, que são unidades de informação de controle e trechos de mensagens enviados pelas estações de transporte, utilizando-se do

campo de dados dos pacotes do nível 3 do X.25. Foram utilizados 10 (dez) tipos de fragmentos, que podem ser divididos em dois grupos. Fragmentos para estabelecimento e encerramento de conexões e fragmentos para transferência de dados.

- Fragmentos para estabelecimento e encerramento de conexões:

  . PCONEX (PEDIDO DE CONEXAO)
  . CONACEITA (ACEITA CONEXAO SOLICITADA)
  . REJCON (REJEITA CONEXAO SOLICITADA)
  . ACKAC (CONFIRMA ACEITACAO DE CONEXAO)
  . FCONEX (FECHA CONEXAO)
  . ACKFE (CONFIRMA FECHAMENTO DE CONEXAO)

- Fragmentos para transferência de dados:

  . DADOS (TRANSPORTA DADOS)
  . INTERRUPCAO (TRANSPORTA INTERRUPCAO)
  . TELEGRAMA (TRANSPORTA TELEGRAMA)
  . ACKDADOS (CONFIRMA RECEBIMENTO DE DADOS)

Uma descrição mais detalhada acerca dos serviços providos por este protocolo de Transporte, bem como dos serviços por ele requeridos ao nível de Rede pode ser obtida na referência [1]. Uma descrição mais detalhada do protocolo de Transporte em si, contendo descrição dos fragmentos, tabela de procedimentos, decrição dos estados e temporizações envolvidas pode ser conseguida na referência [6].

Após reuniões ocorridas entre representantes de ambas as universidades, determinou-se que a implementação inicial deveria ser simplificada de maneira a prover apenas os serviços básicos ao nível superior, podendo, evidentemente, ser expandida no futuro. Entre os serviços não implementados nesta versão preliminar, encontram-se multiplexação de circuitos virtuais, recuperação de falhas, controle de erros fim a fim e serviço de Telegramas. Determinou-se também que uma nova especificação, ao menos informal, deveria ser elaborada, cabendo à UFPE a especificação dos serviços (providos pelo protocolo de Transporte e requeridos do nível de Rede [1]) e à UFPB a especificação do Protocolo de Transporte propriamente dito [6].

Vale a pena salientar que, para ser possível a utilização dos serviços do conversor X.25 (o qual será fornecido pela EMBRATEL às universidades envolvidas), foi necessário o desenvolvimento e implementação de uma interface entre este conversor e a Estação de Transporte [5].

## 4. IMPLEMENTACAO DA ESTACAO DE TRANSPORTE NA UFPE

### 4.1 Ambiente Computacional da UFPE

A Universidade Federal de Pernambuco dispõe atualmente de um sistema DIGITAL-1091, com 256K palavras de 36 bits, processador central KL10-E, Sistema Operacional por partilha de tempo, TOPS-10 versão 7.01A, contendo um quadro de 11 linguagens de programação de alto nível entre suportadas e não suportadas pelo fabricante.

## 4.2 Comunicação entre Processos no TOPS-10

O Sistema Operacional TOPS-10 dispõe de um software, IPCF (Inter-Process Communication Facility), que permite a comunicação entre JOB's ou Processos, em um mesmo sistema de computação, através da transferência de informações sob a forma de pacotes de dados [2].

Os processos podem solicitar identificadores denominados de PID (Process IDentifier). Tais identificadores são valores únicos no sistema e podem estar associados ao número do JOB ou a um dado nome simbólico especificado pelo Processo. Os processos podem inquirir ao gerenciador de PID's qual o dono de um dado PID, qual o nome simbólico associado a um dado PID ou qual o PID associado a um dado nome simbólico.

A cada processo ativo estão associadas uma fila de envio e uma fila de recepção. Quando um dado processo envia um pacote de dados para outro, este pacote será retirado da fila de envio do processo de origem e será colocado na fila de recepção do processo destinatário. Existem quotas que limitam o número máximo de pacotes enviados e ainda não recebidos.

As operações básicas permitidas são: envio de um pacote de dados, recuperação do primeiro pacote da fila de recepção e consulta à fila de recepção (qual o tamanho da fila ?, qual o tamanho do primeiro pacote da fila ?, etc.). Os pacotes IPCF podem ser formados em 'SMALL MODE' ou 'PAGE MODE'. Pacotes em 'SMALL MODE' têm o tamanho do campo de dados limitado a 10 palavras de 36

bits. Pacotes em 'PAGE MODE' têm o tamanho do campo de dados limitado a 1 página de 512 palavras.

## 4.3 A Linguagem de Programação escolhida

Apesar de não ser suportada pelo fabricante, a linguagem PASCAL seqüencial foi escolhida para nossa implementação por duas razões. Em primeiro lugar por ser uma linguagem estruturada, de grande popularidade e fácil aceitação, e por permitir o uso intensivo de tipos abstratos de dados. Em segundo lugar, pelo fato da implementação PASCAL atualmente disponível no DEC-10 da UFPE [3] possuir uma interface com o TOPS-10 bastante poderosa, haja vista que é possível se invocar praticamente todas as chamadas ao monitor (MONITOR CALLS) diretamente através de um programa PASCAL.

Vale também ressaltar que existe um software de suporte disponível para o PASCAL que objetiva tornar transparente ao usuário a forma como são obtidos os serviços fornecidos pelo IPCF. Deste modo, quando se deseja, por exemplo, enviar um pacote em 'PAGE MODE' não é necessário especificar o endereço da página a ser transmitida. Basta apenas passar como parâmetro o registro (PASCAL) ou estrutura equivalente, a ser enviado.

## 4.4 Estruturas Básicas de Dados

Dentre as estruturas de dados utilizadas na implementação da Estação de Transporte, vale salientar as seguintes: Bloco de Controle de Ligação, Bloco de Espera de Ligação, Mapa de Circuitos Virtuais e Mapa de Portas.

O Bloco de Controle de Lisacão consiste de uma entrada para cada conexão ativa, contendo basicamente os seguintes campos: estado da lisacão, canal lógico associado à lisacão, PID do processo usuário local, encaminhacão da lisacão, porta local, porta destino, créditos cedidos, créditos recebidos, número de seqüência da transmissão e recepção, classe da lisacão e opcionais requeridos, nome dos processos local e remoto, tamanhos máximos das mensagens de transmissão e recepção, ponteiros para os buffers de dados alocados, entre outros.

O Bloco de Espera de Lisacão consiste de uma entrada para cada pedido de espera de lisacão, contendo, basicamente os seguintes campos: PID do processo local, porta cativa , classe solicitada e opcionais exigidos, tamanhos máximos das mensagens de transmissão e recepção, entre outros.

O Mapa de Circuitos Virtuais consiste de uma entrada para cada circuito virtual possível (em nossa implementacão é limitada ao número de portas do conversor X.25), contendo os seguintes campos: estado (livre ou ocupado), ponteiro para a entrada correspondente no Bloco de Controle de Lisacão e endereco do HOST remoto (ETD + REDE).

O Mapa de Portas consiste de uma entrada para cada porta possível contendo um ponteiro para a entrada correspondente no Bloco de Controle de Lisacão ou um ponteiro para a entrada correspondente no Bloco de Espera de Lisacão.

## 4.5 Interface com os Níveis Adjacentes

Em virtude de dispormos de um software da natureza do IPCF [2], resolvemos optar por implementar um modelo de interface baseado na troca de primitivas nos dois sentidos. No nosso modelo, existem basicamente três tipos de primitivas: primitivas de solicitação de serviço (ABRECON, ESPERACON, ESTABCV, etc.), primitivas de resposta a solicitações de serviços (RESP_ABRECON, RESP_ESPERACON, RESP_ESTABCV, etc.) e primitivas de indicação (CHEG_MENSAGEM, CHEG_INTERRUPCAO, IND_ESTABCV, etc.). As primitivas de indicação têm o objetivo de permitir que um dado nível possa notificar, ao nível superior, a ocorrência de um dado evento (estabelecimento de circuito virtual, encerramento de conexão, aborto de conexão, chegada de mensagem, entre outros) imediatamente, isto é, sem a necessidade de ser invocada qualquer primitiva pelo nível superior. A figura 4.5.1 ilustra o uso destes tipos de primitivas, tomando como exemplo o serviço de encerramento unilateral de conexões.

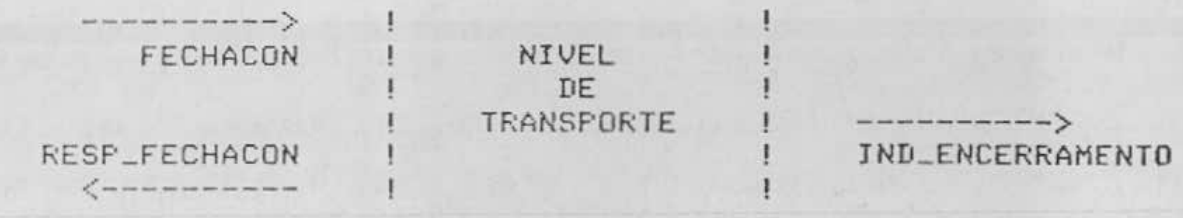


fig. 4.5.1 - serviço de encerramento de conexões
(primitivas FECHACON, RESP_FECHACON e IND_ENCERRAMENTO)

As primitivas são construídas da seguinte forma: o primeiro campo é destinado ao código de identificação da primitiva, e nos demais campos são passados os parâmetros que a compõem. Cada primitiva pode ser enviada sob a forma de pacotes de dados IPCF em 'SMALL MODE' ou 'PAGE MODE', dependendo de seu tamanho. Quando se for transmitir uma dada primitiva, deve-se passar como parâmetros o código da primitiva, o tamanho e o endereço do registro que contém seus campos.

Na recepção das primitivas, recupera-se o primeiro pacote da fila de IPCF e verifica-se se esta é uma primitiva válida; se for, recupera-se os demais parâmetros em um registro global disponível para tratamento; do contrário, descarta-se o pacote recebido.

O endereçamento dos processos comunicantes é feito através do uso dos PID's do IPCF. Desta forma, quando um processo invocar, por exemplo, um ABRECON, a Estação de Transporte (que possui um PID associada a um nome simbólico conhecido), deverá guardar o PID do processo a fim de poder responde-lhe posteriormente. A desativação (proposital ou não) da Estação de Transporte, do processo que efetua a interface com o conversor X.25 (daqui por diante chamado de NIVEL-3) ou de algum processo usuário local pode ser detectada quando for enviado, ao processo que foi desativado, um pacote IPCF, pois, após sua desativação, o PID do processo passa a inexistir.

As primitivas utilizadas para efetuar a comunicação com o NIVEL-3, classificadas em primitivas para o gerenciamento de circuitos virtuais e primitivas para transferência de dados, serão listadas a seguir.

- Primitivas para o gerenciamento de circuitos virtuais:

. ESTABCV (ESTABELECA CIRCUITO VIRTUAL)

. RESP_ESTABCV (RESPOSTA A SOLICITACAO DE C. V.)

. IND_ESTABCV (INDICACAO DE ESTABELECIMENTO DE C. V.)

. LIMPACV (ENCERRE CIRCUITO VIRTUAL)

. RESP_LIMPACV (RESPOSTA AO PEDIDO DE ENCERRAMENTO DE C. V.)

. IND_FIMCV (INDICACAO DE ENCERRAMENTO DE C. V.)

. RESETCV (RESET EM CIRCUITO VIRTUAL)

. RESP_RESETCV (RESPOSTA A SOLICITACAO DE RESET)

. IND_RESET (INDICACAO DE RESET EM C. V.)

. RESTART (SOLICITACAO DE RESTART)

- Primitivas para transferência de dados:

. TRPACOTE (TRANSMITA PACOTE)

. RESP_TRPACOTE (RESPOSTA AO PEDIDO DE TRANSMISSAO)

. CHEG_PACOTE (INDICACAO DE CHEGADA DE PACOTE)

. IND_CVLIVRE (CIRCUITO VIRTUAL DESCONGESTIONADO)

A primitiva IND_CVLIVRE foi incluída com o objetivo de efetuar o controle de fluxo entre a Estação de Transporte e o NIVEL-3. Quando a Estação de Transporte, ao solicitar o envio de um pacote (através da primitiva TRPACOTE), receber uma resposta indicando que a fila do circuito virtual correspondente está relativamente cheia, ela deverá suspender a transferência de dados naquele circuito virtual até que receba uma primitiva IND_CVLIVRE.

As primitivas utilizadas para efetuar a comunicação com o(s) nível(is) superior(es), classificadas em primitivas de estabelecimento e encerramento de conexões e primitivas de

transferência de dados, serão listadas a seguir.

- primitivas de estabelecimento e encerramento de conexões:

- ABRECON (ABRE CONEXAO)
- RESP_ABRECON (RESPOSTA AO PEDIDO ABERTURA DE CONEXAO)
- ESPERACON (ESPERA CONEXAO)
- RESP_ESPERACON (RESPOSTA AO PEDIDO DE ESPERA DO CONEXAO)
- CANCELESP (CANCELE PEDIDO DE ESPERA DE CONEXAO)
- FECHACON (FECHE CONEXAO UNILATERALMENTE)
- RESP_FECHACON (RESPOSTA AO PEDIDO DE FECHAMENTO DE CONEXAO)
- IND_ENCARRAMENTO (INDICACAO DE ENCERRAMENTO REMOTO)
- ABORTACON (ABORTE A CONEXAO)
- RESP_ABORTACON (RESPOSTA AO PEDIDO DE ABORTO)
- IND_ABORTO (INDICACAO DE ABORTO DE CONEXAO)

- Primitivas de transferência de dados:

- ENVMSG (ENVIE MENSAGEM)
- RESP_ENVMSG (RESPOSTA AO PEDIDO DE ENVIO DE MENSAGEM)
- CHEG_MENSAGEM (INDICACAO DE CHEGADA DE MENSAGEM)
- ENVINT (ENVIE INTERRUPCAO)
- RESP_ENVINT (RESPOSTA AO PEDIDO DE ENVIO DE INTERRUPCAO)
- CHEG_INTERRUPCAO (INDICACAO DE CHEGADA DE INTERRUPCAO)
- IND_DESOCUPADO (CONEXAO DESCONGESTIONADA)

A primitiva IND_DESOCUPADO foi introduzida com a finalidade de efetuar o controle de fluxo entre a Estação de Transporte e o(s) nível(is) superior(es). Quando, ao invocar a primitiva ENVMSG, um processo usuário local receber uma resposta negativa indicando que a conexão está temporariamente congestionada, o processo terá que

suspender a transferência de dados (exceto interrupção) nesta conexão até que receba uma primitiva IND_DESOCUPADO.

As primitivas utilizadas para efetuar a interface com o temporizador foram LIGA_TEMPORIZADOR e ESTOURO_TEMPORIZADOR.

4.6 Os Estados da Estação

Em nossa implementação, cada ligação pode se encontrar em um dos estados abaixo citados:

INATIVO: este é um estado fictício, onde ainda não há ligação estabelecida. Serve apenas para indicar que esta entrada do bloco de ligação está disponível.

ESPERANDO_CV: neste estado, a Estação de Transporte Já recebeu um ABRECON de um processo usuário local, os parâmetros já foram criticados, foi enviado um ESTABCV ao NIVEL-3 e está sendo aguardada uma resposta.

LIGANDO_ATIVO: foi obtida uma resposta positiva do NIVEL-3, foi enviado através do circuito virtual estabelecido um fragmento PCONEX à Estação de Transporte remota (embutido numa primitiva TRPACOTE) e está sendo esperada uma primitiva CHEG_PACOTE que traga como dados um fragmento de resposta (CONACEITA ou REJCON).

LIGANDO_PASSIVO: a Estação de Transporte local recebeu o fragmento PCONEX e havia algum processo local disposto a estabelecer conexão. Foi enviado um CONACEITA à Estação de Transporte remota e está sendo esperado um ACKAC.

DADOS: a conexão foi estabelecida e permanece ativa a transferência de dados nos dois sentidos até que as estações local e/ou remota manifestem o interesse em encerrá-la.

ABORTANDO: alguma condição anormal aconteceu, tal como falha na Estação de Transporte local, falha na Estação de Transporte remota, RESET, congestionamento, processo local ou remoto foi desativado, não há memória disponível localmente, etc. Foi enviado um fragmento FCONEX com marca de aborto. Praticamente todos os recursos alocados foram liberados e está sendo esperado um fragmento ACKFE.

DESLIGANDO_LOCAL: o processo local solicitou o fechamento unilateral da conexão por não possuir mais dados a serem transmitidos. Os recursos alocados para o envio de dados foram liberados. Foi enviado um FCONEX à Estação de Transporte remota e está sendo aguardado um ACKFE.

DESLIGADO_LOCAL: a resposta ao FCONEX enviado chegou e agora apenas o processo remoto poderá transmitir dados através da conexão.

DESLIGANDO_SIMULTANEO: ambos os processos solicitaram o encerramento da conexão. Praticamente todos os recursos alocados foram liberados. Foi enviado um ACKFE à Estação de Transporte remota e está sendo esperado o mesmo fragmento.

DESLIGADO_REMOTO: o processo remoto solicitou o encerramento unilateral da conexão. Os recursos alocados para a recepção de dados remotos foram liberados. Foi enviado um fragmento ACKFE à

Estação de Transporte remota.

### 4.7 Software em Baixo nível utilizado

Algumas rotinas tiveram que ser implementadas em baixo nível (ASSEMBLY) devido às restrições da linguagem PASCAL. Dentre estas rotinas destacam-se a rotina que retorna em uma dada variável o endereço de outra variável ou registro, e as rotinas de compactação e descompactação dos campos dos fragmentos (palavra inteira de 36 bits para byte de 8 bits, ARRAY OF BOOLEAN para byte, CHAR para byte e as operações inversas).

### 4.8 Alocação de Buffers

O método de alocação adotado foi o de pré-alocação de buffers para os créditos cedidos visando a maior mensagem, isto é, tenta-se alocar previamente tantos buffers quantos forem necessários para receber a mensagem de maior tamanho (determinada no estabelecimento da conexão). Como o tamanho do campo de dados do fragmento DADOS é 117 bytes, o número de buffers necessários para a recepção da mensagem de tamanho máximo é, logicamente, o tamanho da maior mensagem dividido por 117 (mais 1, em caso de quebra). Entretanto se não for possível esta alocação prévia, periodicamente se tenta alcançar este objetivo através de alocação dinâmica.

### 4.9 Inicialização da Estação de Transporte

A Estação de Transporte , bem como o NIVEL-3, serão ativados automaticamente, como se fossem módulos hibernantes do sistema TOPS-10, quando da carga do sistema.

Dentre as tarefas efetuadas pela Estação de Transporte quando da sua inicialização, encontram-se: verificação de ativação do gerenciador de PID's; verificação da ativação do NIVEL-3; verificação de ativação do Temporizador; solicitação de um PID ao gerenciador e associação a um nome simbólico conhecido; envio de RESTART ao NIVEL-3; inicialização das estruturas de dados. Após a ativação da Estação de Transporte, o operador deverá colocar os processos servidores em espera de conexão em suas respectivas portas ativas.

4.10 Comunicação com o Operador

A intervenção do operador será necessária quando ocorrer uma falha irrecuperável da Estação de Transporte ou quando sua ativação estiver temporariamente impossibilitada.

Após ocorrida uma falha irrecuperável, será enviada uma mensagem de erro para todos os JOBs em modo operador do sistema, inclusive para a console. Será necessária a reinicialização da Estação de Transporte.

Se por acaso, quando estiver sendo inicializada a Estação de Transporte, algum de seus servidores não estiver ativo, será enviada uma mensagem descritiva ao(s) operador(es) e a Estação de Transporte ficará esperando uma resposta (continue ou pare). Caberá ao operador ativar o devido servidor (se possível) e determinar a continuação da Estação de Transporte. Os processos servidores da Estação de Transporte são o gerenciador de PIDs, o NIVEL-3 e o Temporizador.

As mensagens de erro a serem enviadas ao operador são basicamente compostas de 3 campos: código identificador da mensagem, código de erro retornado na chamada ao monitor mal sucedida e descrição sucinta do erro ocorrido.

4.11 Erros na Implementação

Existem basicamente três tipos de erro possíveis de ocorrer durante o funcionamento da Estação de Transporte: erros endereçados ao operador (os quais já foram discutidos no item 4.10), erros de procedimento e erros de primitivas.

Os erros de procedimento são erros que visam à detecção de possíveis falhas na implementação do protocolo. Os erros de procedimento dos quais nossa implementação for responsabilizada serão gravados em um arquivo de LOG, juntamente com data e hora de ocorrência e todo o contexto da ligação em questão.

Os erros de primitivas são erros passados como código de retorno nas primitivas de resposta, devido a alguma inconsistência de parâmetro, falta de recurso, impossibilidade remota, etc. Quando o erro se origina na Estação de Transporte remota, ele será passado ao processo local como um valor negativo. Do contrário será entregue como um valor positivo.

4.12 O Temporizador

O nosso temporizador foi desenvolvido visando à utilização do mesmo não só pela Estação de Transporte mas também pelo NIVEL-3 e por qualquer processo do nível superior que o necessitar. O

temporizador consiste de um processo cuja ativação, inicialização e comunicação com o operador são semelhantes à Estação de Transporte. A interface do Temporizador com qualquer outro usuário se dá através da troca de primitivas via envio e recepção de pacotes IPCF.

As solicitações de temporização devem ser feitas através do envio de uma primitiva LIGA_TEMPORIZADOR, a qual é composta dos seguintes campos: código da primitiva, número de Unidades Mínimas de Tempo a serem computadas, tipo de temporização, identificador de conexão e encarnação do temporizador. O campo de encarnação foi introduzido pelo fato de que não é viável deslisar um dado temporizador, porque o tempo gasto para efetuar tal operação (envolve tempo gasto com o Temporizador e com o usuário) seria maior do que se o estouro do temporizador fosse simplesmente ignorado. (isto pode ser consegüido mudando-se a encarnação do temporizador, pois as indicações de 'time-out' atrasadas não serão levadas em consideração por não coincidirem com a encarnação atual.) Além do mais em casos de congestionamento intensivo de pacotes IPCF, poder-se-ia chegar a inconsistências graves.

O temporizador notificará aqueles usuários cuja temporização já foi esgotada através do envio de uma primitiva ESTOURO_TEMPORIZADOR, a qual é composta dos seguintes campos: tipo de temporização, identificador da conexão e encarnação do temporizador.

Na Estação de Transporte utilizamos a temporização de inatividade remota, temporização de solicitação de circuito virtual e temporização de espera de conexão. A temporização de inatividade remota consiste de um tempo máximo, fornecido pelo processo local, que a Estação de Transporte suportará sem receber fragmentos da Estação de Transporte remota. Passado este tempo, a conexão será abortada. A temporização de solicitação de circuito virtual consiste de um tempo máximo que a Estação de Transporte tolerará sem receber resposta à solicitação de circuito virtual. Passado este tempo, será dada uma resposta negativa ao processo local. A temporização de espera de conexão é um tempo máximo, fornecido pelo processo local, que determina o limite de tolerância sem estabelecimento de conexão. Esgotado este tempo, o ESPERACON será desativado, e uma resposta negativa será dada ao processo local.

4.13 O Esqueleto da Estação de Transporte

O corpo principal do programa que implementa a Estação de Transporte é o abaixo apresentado.

```
BEGIN
   inicialize_estação_de_transporte;
   WHILE      toda_vida      DO
   BEGIN
      receba_próxima_primitiva;
      CASE      primitiva      OF
         ABRECON:  trata_ABRECON;
         ESPERACON:  trata_ESPERACON;
         CANCELESP:  trata_CANCELESP;
         FECHACON:  trata_FECHACON;
         ABORTACON:  trata_ABORTACON;
         ENVMSG:  trata_ENVMSG;
         ENVINT:  trata_ENVINT;
         RESP_ESTABCV:  trata_RESP_ESTABCV;
         RESP_LIMPACV:  trata_RESP_LIMPACV;
         IND_FIMCV:  trata_IND_FIMCV;
         RESP_RESETCV:  trata_RESP_RESETCV;
```

```
            IND_RESET:  trata_IND_RESET;
            RESP_TRPACOTE:  trata_RESP_TRPACOTE;
            CHEG_PACOTE:  trata_CHEG_PACOTE;
            IND_CVLIVRE:  trata_IND_CVLIVRE;
            ESTOURO_TEMPORIZADOR:  trata_ESTOURO_TEMPORIZADOR;
        END;
    END;
END.
```

A invocação da rotina de recepção de primitiva é feita de maneira que a estação fique bloqueada (em estado HIBER) até que chegue um pacote de IPCF em sua fila de recepção. Dependendo da primitiva recebida, será chamada a rotina de tratamento de recepção específica.

4.14 Testes da Estação

A ferramenta de testes até agora utilizada é bastante rudimentar, o que não nos permite considerá-la concluída. Trata-se de um programa, com pequena inteligência, que pode simular o funcionamento da Estação de Transporte, do NIVEL-3 ou de processos usuários locais. Este programa é alimentado com cenários via terminal de vídeo. Ativamos um processo para simular a Estação de Transporte, outro para simular o NIVEL-3 e alguns processos para simular a existência de processos servidores e consumidores. Toda primitiva enviada ou recebida por quaisquer destes processos é mostrada no vídeo. Existem operações para consultar todas as tabelas e estruturas utilizadas. Tão logo sejam fornecidos pela EMBRATEL o conversor e sejam instalados os nós REXPAC, teremos condições de aplicar uma ferramenta mais poderosa e confiável, possibilitando a viabilização dos testes integrados.

## 5. CONSIDERACOES FINAIS E DIRECOES FUTURAS

Apesar das dificuldades enfrentadas, as quais envolveram desde problemas com a implementação PASCAL disponível no DEC-10 da UFPE até a impossibilidade momentânea de efetuar os teste globais (em virtude de ainda não dispormos dos equipamentos combinados), apesar de nossa implementação ter sido bastante simplificada (em virtude de o prazo de conclusão do projeto ser bastante curto), acreditamos que a experiência foi válida e nos proporcionou resultados significativos tanto no tocante aos conhecimentos básicos adquiridos como no ferramental desenvolvido, de bastante utilidade para os experimentos futuros.

Está previsto, na continuação dos nossos trabalhos ligados a este projeto, além da elaboração de uma metodologia que possibilitará a efetuação dos testes integrados, uma especificação formal do protocolo e dos serviços. Pretendemos também otimizar nossa implementação da Estação de Transporte, incluindo novos serviços que torná-la-ão mais confiável e eficiente, tais como multiplexação de circuitos virtuais, controle de erros fim a fim, recuperação de falhas, serviço de telegramas, controle de fluxo mais apurado entre os níveis, entre outros.

AGRADECIMENTOS

Queremos manifestar nossos agradecimentos à EMBRATEL e à Secretaria Especial de Informática (SEI), pelo suporte financeiro concedido, e ao Núcleo de Processamento de Dados da UFPE, pelo suporte técnico fornecido, ambos de suma importância para o desenvolvimento deste projeto.

REFERENCIAS:

[1] Cunha, P.R.F.; Monteiro, J.A.S.; Lucena, E.B. - "Especificação dos Serviços do Protocolo de Transporte para a Rede CEPINNE"; Relatório Técnico submetido à EMBRATEL, dezembro 1984.

[2] DIGITAL Equipment Corporation - "Communication Between Processes: IPCF"; Software Notebook 4, July 1982 (pp. 8-1 - 8-24).

[3] Kisicki, E.; NAGEL, H. - "PASCAL for the DECsystem-10"; novembro 1976.

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

[5] Monteiro, J.A.S.; Cunha, P.R.F.; Rodrigues, J.S.F. - "Uso de um Conversor "START/STOP" - X.25 para o Acesso de Computadores a Redes Públicas"; submetido ao 3o. Simpósio Brasileiro de

Redes de Computadores, Janeiro 1985.

[6] Sauvé, J.P. et al. - 'Especificação do Protocolo de Transporte para a Rede CEPINNE'; Relatório Técnico submetido à EMBRATEL, dezembro 1984.

[7] Secretaria Especial de Informática - SEI - 'Projeto CEPINNE'; Janeiro 1981.

[8] Tanenbaum, A.S. - 'Computer Networks'; Prentice-Hall, Inc., Englewood Cliffs, New Jersey, Estados Unidos, 1981, 515 pp.

[9] Day, J.D.; Zimmermann, H. - 'The OSI Reference Model'; Proceedings of the IEEE, volume 71, número 12, Dezembro 1983, (pp. 1334-1340).