Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica

Projeto e Desenvolvimento de uma Interface de Rede Local para Aplicações em Automação Industrial

Autor : Leonardo Appezato
Orientador : Prof. Dr. Maurício Ferreira Magalhães

Tese apresentada à Faculdade de Engenharia Elétrica da UNICAMP como parte dos requisitos exigidos para a obtenção do Título de Mestre em Ciências.

Junho de 1990

Este exemplar corresponde à redação final da defendida por LEONARDO APPEZATO
e aprovada pela Comissão Julgadora em 13 / 07 / 1990.

Orientador

À Bia, Léo e Rafa.

Agradecimentos

Ao Mário Bento de Carvalho, Rubens Campos Machado e Roberto Fernandes Tavares Filho, respectivamente Diretor, Chefe do Departamento DCP e Chefe da Divisão ETR, do Instituto de Automação do Centro Tecnológico para Informática, pela possibilidade de realizar o projeto que originou este trabalho.

Aos "ex-técnicos" Marcelo e Reinaldo, pelo trabalho de montagem das placas dos protótipos.

Ao Hélio Azevedo, pelo auxílio no uso dos equipamentos TEKTRONIX e nos "apuros" com a depuração dos programas.

Ao Maurício de Faria, pela ajuda na retirada dos "bugs" da placa PC.

Ao pessoal que integra ou já integrou o DCP: Koyama, Olga, Edeneziano, Isaías, Hélio, Hexsel, Beiral, Cidinha, Juziani, Clarinda (UDD), Paulo, Takao, Eliane, Juan e Aqueo.

Ao Maurício Magalhães, pela orientação paciente e pela amizade.

Resumo

Esta dissertação apresenta o projeto de uma interface de rede local, desenvolvida no Instituto de Automação do Centro Tecnológico para Informática. Este projeto tem uma proposta de implementação de hardware e software objetivando sua utilização em aplicações industriais.

O projeto constitue-se em uma placa inteligente onde se localizam as duas camadas inferiores do Modelo de Referência OSI da ISO, implementadas de uma forma similar à especificação Mini-MAP. A subcamada MAC é do tipo "token-bus" com taxa de transmissão de 1Mbit/seg e permitindo até 254 estações no mesmo segmento de rede. O "chip" controlador da rede local utilizado não é aderente ao padrão IEEE 802.4. A subcamada LLC opera em Classe 3 segundo o padrão IEEE 802.2, suportando prioridade no processamento das mensagens.

Os programas existentes na interface de rede local operam utilizando-se dos serviços de um núcleo multi-tarefas. Tanto os programas como o núcleo multi-tarefas são escritos na linguagem de programação C.

A interface de rede local possui duas versões: uma para ser utilizada em barramentos do tipo IBM PC/XT e outra para o barramento ECB.

Abstract

This dissertation presents the project of a Local Area Network (LAN) Interface developed at the Instituto de Automação of the Centro Tecnológico para Informática. This project has a implementation proposal of hardware and software for industrial application needs.

The project constitutes of a intelligent board where the two lower layers of the OSI Reference Model reside, which are implemented similar to the Mini-MAP specification. The MAC sublayer is a "token-bus" type with a transmission rate of 1Mbit/s, allowing up to 254 nodes in the same LAN segment. The LAN controller chip used is not compatible to the IEEE 802.4 Standard. The LLC sublayer operates in Class 3 of the IEEE 802.2 Standard, supporting priority in the processing of messages.

The programs running in the LAN Interface use the services of a multi-task kernel. This programs and the kernel are written using the C programming language.

The LAN Interface has two versions: one for the IBM PC/XT bus and other for the ECB bus.

Conteúdo

Capítulo 1

Introdução

Há alguns anos as redes de computadores não passavam de ferramentas exóticas de pesquisa, utilizadas por alguns poucos especialistas. Atualmente, é difícil de se encontrar computadores, quer sejam computadores pessoais ou supercomputadores, que não estejam conectados à alguma rede de comunicação. A utilização de correio eletrônico, por exemplo, é uma realidade diária para milhões de pessoas em todo o mundo [TANE89].

A queda contínua dos custos de hardware, seguida pelo aumento do desempenho e recursos dos computadores (os microprocessadores atuais podem ter velocidade e recursos de memória comparáveis aos de um minicomputador), têm provocado grandes alterações na forma pela qual uma organização coleta, processa e utiliza informações. Verifica-se um aumento no uso de estações de trabalho destinadas a uma única função, tornando-as sistemas mais "amigáveis" e acessíveis aos usuários. Entretanto, o barateamento do custo de hardware diminue o ciclo de vida do mesmo, agravando os problemas de conversão do software. Os custos de conversão do software podem ser reduzidos pela decomposição dos grandes sistemas de computadores em componentes de menor porte e separados [STAL84].

Por outro lado, ainda que o custo do hardware de processamento de dados tenha caído bruscamente, o mesmo não tem ocorrido com o custo dos equipamentos eletro-mecânicos essenciais, tais como memória de massa e impressoras de linha, o qual tem se mantido alto. Com a dispersão do sistema computacional, esses dispositivos precisam ser compartilhados de alguma forma.

As vantagens de interligação dos equipamentos dispersos em uma organização, quer seja para a troca de dados entre eles ou para o compartilhamento dos periféricos de alto custo, fizeram com que surgissem as *redes locais de computadores.*

Uma rede local é um sistema para a interconexão de computadores, terminais ou equipamentos periféricos, que possibilita a comunicação entre eles dentro de um ambiente local como, por exemplo, um conjunto de prédios, um campus universitário ou um complexo industrial, onde os dispositivos se acham separados por distâncias que não excedem alguns poucos quilômetros. As interfaces físicas e os protocolos das redes locais são projetadas visando uma operação eficiente para curtas distâncias, utilizando meios de alta qualidade (por

exemplo, cabos coaxiais, par trançado blindado ou fibras ópticas), resultando em baixas taxas de erro de transmissão (da ordem de um erro em 10^8 bits). As taxas de transmissão de dados são altas (acima de 1 megabits por segundo), permitindo que muitas estações (da ordem de 200) possam compartilhar um único meio de comunicação.

Na prática, as redes locais se apresentam com uma grande diversidade em relação às suas topologias, meios de transmissão e protocolos de comunicação. Não existe uma solução única que satisfaça os requisitos operacionais de todas as aplicações existentes [MEND89]. Dentre as principais aplicações para as redes locais pode-se salientar as seguintes: redes para centros de processamento de dados, para escritórios comerciais e técnicos, para computadores pessoais e para a comunicação fabril.

Esta dissertação apresenta o desenvolvimento de uma *Interface de Rede Local* (IRL) com características apropriadas para a aplicação em comunicação fabril. Este projeto foi desenvolvido no Instituto de Automação do Centro Tecnológico para Informática (CTI) e configurou-se no primeiro passo com o objetivo de adquirir conhecimentos nessa área. O desenvolvimento da IRL constitue-se na implementação de uma placa inteligente executando os protocolos de comunicação referentes às duas camadas inferiores do Modelo de Referência OSI da ISO. A implementação do protocolo de comunicação foi feita, onde possível, utilizando-se padrões internacionais. Este projeto faz parte de um outro mais amplo, em andamento no CTI [APPE89], que tornará disponível o protótipo de um nó de comunicação com características semelhantes às da arquitetura Mini-MAP, definida na especificação MAP ("Manufacturing Automation Protocol").

O restante deste texto encontra-se organizado na forma descrita a seguir. O Capítulo 2 apresenta um histórico sobre a criação do Modelo de Referência OSI da ISO, assim como sua arquitetura básica de sete camadas. O Projeto 802 da IEEE é também citado por definir uma arquitetura adotada na implementação de redes locais. Este capítulo aborda, ainda, as redes locais utilizadas na automação industrial, suas vantagens e os esforços na utilização de padrões internacionais, conduzidos pela General Motors americana através da especificação MAP. Algumas das principais características dos produtos MAP disponíveis no mercado internacional são discutidas.

O Capítulo 3 apresenta a subcamada LLC, com destaque para a sua operação Tipo 3, por se tratar do escopo principal da implementação dos protocolos de comunicação neste projeto.

O Capítulo 4 apresenta o projeto da IRL, abordando a filosofia de implementação do hardware e do software. As peculiaridades do projeto e a estruturação do software são discutidas em detalhe. No final deste capítulo é dado um exemplo de operação dos programas executados na IRL e são apresentados alguns dados de desempenho da implementação, seguidos de comentários.

O Capítulo 5 apresenta as conclusões, onde são discutidas formas de se aprimorar o trabalho e sugestões para projetos futuros.

Capítulo 2

Redes de Computadores

A indústria de computadores, ainda que recente, tem tido um grande progresso ao longo dos últimos anos. Nas suas primeiras décadas de existência os sistemas computacionais se caracterizaram pelo alto grau de centralização, ocupando normalmente uma única sala com grande espaço físico. Vinte anos após, o surgimento de microcomputadores igualmente poderosos aos grandes computadores de então, aliado ao desenvolvimento da tecnologia de comunicação, influenciaram na forma pela qual os sistemas computacionais são organizados. O conceito de um único computador de grande porte realizar todo o trabalho e o de que os usuários devem levar seus trabalhos até o computador (ao invés de se levar os computadores até os usuários), está sendo substituído pelo conceito de se deixar que vários computadores, interconectados por uma rede de comunicação, façam o mesmo trabalho.

Porém, o desenvolvimento de redes de computadores feito até agora tem gerado várias "caixas pretas". Cada fabricante de computador possue sua própria arquitetura de rede, incompatível com a de qualquer outro fabricante. Esta situação deve ser alterada, uma vez que praticamente todas as indústrias de computadores concordaram recentemente na adoção de padrões internacionais que descrevem a arquitetura de redes. Esses padrões são conhecidos como Modelo de Referência OSI ("Open Systems Interconnection") da ISO ("International Standards Organization"). Com isso, espera-se que no futuro próximo as redes proprietárias dêem lugar às redes padronizadas, permitindo a comunicação entre computadores de diferentes fabricantes com um mínimo de esforço e estimulando ainda mais a utilização de redes [TANE89].

2.1 Modelo de Referência OSI

Em 1978, o Comitê Técnico 97 (TC 97) para Processamento de Informações da ISO, reconhecendo que a existência de padrões para redes de sistemas heterogêneos era necessária urgentemente, criou um novo subcomitê (SC 16) para a "interconexão de sistemas abertos". O termo *sistema aberto* foi escolhido para enfatizar que em conformidade com padrões OSI,

um sistema estaria aberto para se comunicar com qualquer outro sistema que utilizasse os mesmos padrões [JDAY83].

Depois de 18 meses de trabalho o Modelo de Referência OSI foi criado e enviado ao TC 97 para apreciação. Finalmente, em 1983 ele se tornou um padrão internacional (ISO 7498).

O Modelo de Referência OSI [ISO82] subdivide um sistema de comunicação em camadas. A idéia básica é a de que cada camada aperfeiçoa os serviços fornecidos pelo conjunto de camadas abaixo dela, de tal forma que a camada no topo fornece um conjunto completo de serviços necessários para a execução de aplicações distribuídas.

Um outro princípio básico para a divisão em camadas é a de garantir a independência de cada uma delas através da definição de serviços fornecidos por uma camada para a camada imediatamente superior, sem se importar de como esses serviços são executados. Isso permite que mudanças na forma de operar de uma camada possam ser feitas, contanto que ela ainda forneça os mesmos serviços à camada superior. A técnica é similar à utilizada em programação estruturada onde somente as funções realizadas por um módulo (e não seu funcionamento interno) são conhecidas pelo usuário.

O Modelo de Referência OSI é formado por sete camadas, como ilustra a figura 2.1.

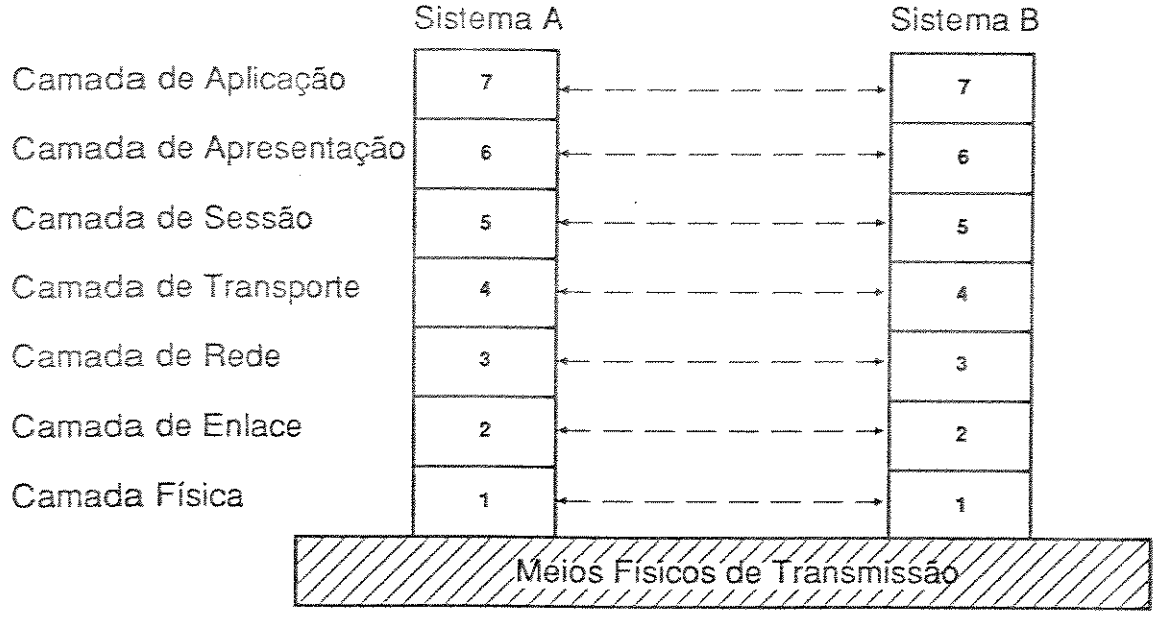

Figura 2.1: Modelo de Referência OSI

Cada camada consiste de um ou mais *subsistemas*, onde cada subsistema contém um ou mais elementos ativos, denominados *entidades* na terminologia OSI [GEPD84].

A camada N (N = 1, 2, ..., 7) de um determinado sistema de comunicação conversa com a camada N de um outro sistema, como ilustra a linha tracejada da figura 2.2.

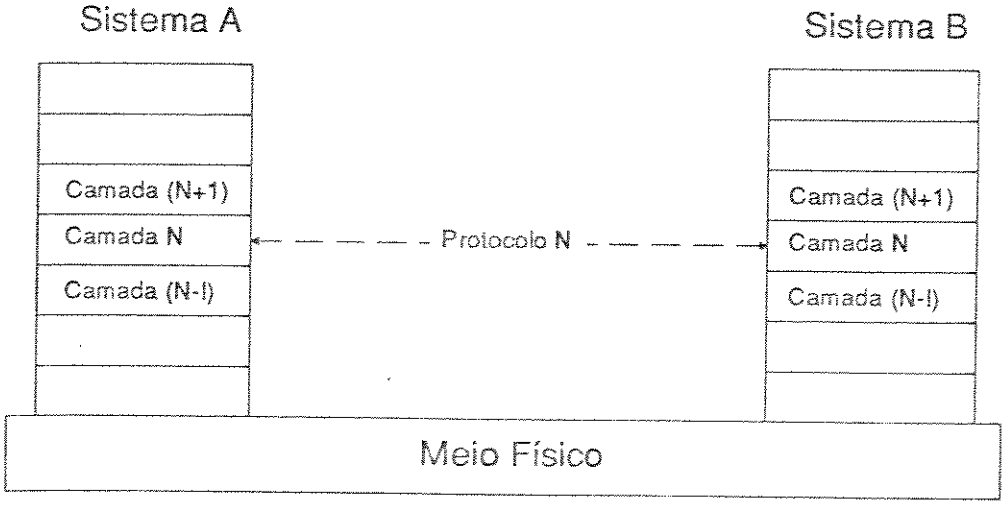

Figura 2.2: Comunicação entre Camadas Pares

As regras e convenções que regem essa conversação são conhecidas por *protocolo da camada N* e os interlocutores dessa conversação são as *entidades da camada N*.

Os dados envolvidos na comunicação entre duas entidades de uma camada N são colocados em uma PDU ("Protocol Data Unit") e passados, no sistema transmissor, para a camada imediatamente inferior e, assim, sucessivamente até alcançar a camada mais baixa (Física), onde a comunicação *física* entre os dois sistemas realmente ocorre (ao contrário das comunicações *virtuais* que ocorrem nas camadas superiores). Do lado do sistema receptor, os dados passam sucessivamente da camada Física para as camadas superiores até atingir a camada N, onde são entregues à entidade destino.

Entre cada par de camadas adjacentes em um mesmo sistema de comunicação, existe uma interface que define as primitivas pelas quais a camada inferior fornece serviços à camada superior. Estes são fornecidos através dos SAPs ("Service Access Point").

Apesar de não explicitar claramente o grau de dispersão geográfica dos sistemas interconectados, o Modelo de Referência OSI é nitidamente orientado para utilização em redes de computadores de longa distância [GIOZ86]. A especificação das três camadas inferiores do modelo não considera importantes características das redes locais, que permitem simplificar as funções de cada uma delas e até mesmo eliminar a necessidade de certas camadas (por exemplo, a camada de Rede). Estão sendo realizados esforços para a adequação do Modelo de Referência OSI à arquitetura de redes locais, devendo-se destacar o trabalho do Comitê 802 da IEEE, que será apresentado no item 2.2.

Uma breve descrição das funções das camadas que compõem o Modelo de Referência OSI é feita a seguir.

Camada Física

Esta camada constitue-se na ligação entre o meio físico (ou seja, a linha de comunicação) e a camada de Enlace de Dados. A camada Física define as características mecânicas, elétricas, funcionais e procedurais relativas ao meio físico da rede, a fim de ativar, manter e desativar conexões físicas para a transmissão de "bits" entre as entidades da camada de Enlace de Dados.

Camada de Enlace de Dados

Esta camada existe para superar as limitações inerentes nos circuitos físicos e para permitir que erros de transmissão possam ser detectados e possivelmente recuperados, ocultando deficiências na qualidade de transmissão. Ela permite, também, que a camada de Rede controle a conexão do caminho de comunicação no meio físico.

Camada de Rede

Existe uma variedade considerável de tipos de instalações e serviços de transmissão de dados disponíveis, tais como: linha comutada, ponto-a-ponto, multi-ponto, circuito virtual, rede de pacotes, etc. A forma na qual os dados são transferidos na rede depende fortemente do tipo de instalação utilizada. Por exemplo, em redes ponto-a-ponto baseadas em serviços de circuitos comutados, os dados são geralmente transferidos intactos, diretamente do emissor para o receptor; em uma rede de comutação de pacotes, entretanto, os dados são divididos em *pacotes* e enviados através de uma rede compartilhada, na qual as conexões físicas entre emissor e receptor são estabelecidas e mantidas por solicitação. A camada de Rede possui a função de tornar essas características distintas de topologias e serviços das redes, transparentes às camadas superiores.

Camada de Transporte

Esta camada é responsável pela transferência transparente de dados entre as entidades da camada de Sessão, de uma forma confiável e viável em termos de custos. A camada de Transporte libera as entidades da camada de Sessão de qualquer preocupação sobre a maneira pela qual os dados são transferidos. Ela também otimiza os serviços da camada de Rede para fornecer o nível de desempenho requerido a um custo mínimo.

A camada de Transporte assume a responsabilidade pela integridade dos dados transferidos através da rede de comunicação. Suas funções incluem controle de sequência fim-a-fim, controle de fluxo, detecção e recuperação de erros e monitoramento da qualidade de serviço fornecida.

Camada de Sessão

Para que dois processos se comuniquem entre si é preciso que uma associação lógica seja estabelecida entre eles, a qual é denominada *sessão*. A camada de Sessão é responsável pelo estabelecimento e terminação de uma sessão, pela organização e sincronização do diálogo entre as duas entidades da camada de Apresentação e pelo gerenciamento da troca de dados entre elas.

Camada de Apresentação

Esta camada é responsável pela garantia da compatibilidade na representação de dados entre os processos de aplicação comunicantes. Dentre as responsabilidades desta camada incluem-se as considerações sobre o formato dos dados (como na formatação de telas ou na saída para impressoras) e transformações de conjuntos de códigos e caracteres.

Camada de Aplicação

A camada de Aplicação fornece ao programa de aplicação (denominado *processo de aplicação* na terminologia OSI) os meios para acessar o ambiente OSI. Esta é a única camada que fornece serviços diretamente ao programa de aplicação e, portanto, todos os serviços OSI utilizados pelos programas de aplicação são fornecidos através da camada de Aplicação.

Para suportar aplicações distribuídas, as funções da camada de Aplicação manipulam informações. Ela fornece gerenciamento de recursos para transferência de arquivos, emulação de terminais e arquivos virtuais e processamento distribuído, dentre outras funções. Esta é a camada que irá conter a maior funcionalidade e é, certamente, a que exigirá a maior variedade de trabalho dos orgãos de padronização.

2.2 Projeto 802 do IEEE

O Projeto 802 desenvolvido pelo IEEE ("Institute of Electrical and Electronics Engineers") dos E.U.A. resultou no estabelecimento de uma arquitetura padrão, nos moldes da arquitetura do Modelo de Referência OSI, com o emprego específico para a implementação de redes locais de computadores [NELS83].

O bom desenvolvimento de um mercado de redes locais somente será possível com a disponibilidade de interfaces de baixo custo. O custo de conexão de um equipamento a uma rede local precisa ser muito menor do que o custo do próprio equipamento. Somando-se este requisito à complexidade dos protocolos das redes locais, conclue-se que a solução para baixar os custos encontra-se no uso da tecnologia VLSI ("Very Large Scale Integration"). Entretanto, os fabricantes de circuitos integrados somente investirão em algo que proporcione um mercado de grandes volumes. A existência de padrões para redes locais garantiriam esses volumes, além do que permitiriam a comunicação entre equipamentos de fabricantes distintos [STAL84].

Com o objetivo de viabilizar rapidamente a produção em grande escala de circuitos integrados e equipamentos, de modo a tornar acessível o uso generalizado da tecnologia de

redes locais, o Projeto 802 do IEEE concentrou-se na elaboração de padrões para as duas camadas inferiores do Modelo de Referência OSI (Física e Enlace de Dados) [GIOZ86]. As peculiaridades da tecnologia associada às redes locais motivaram uma redivisão de serviços e funções em três camadas, como ilustra a figura 2.3.

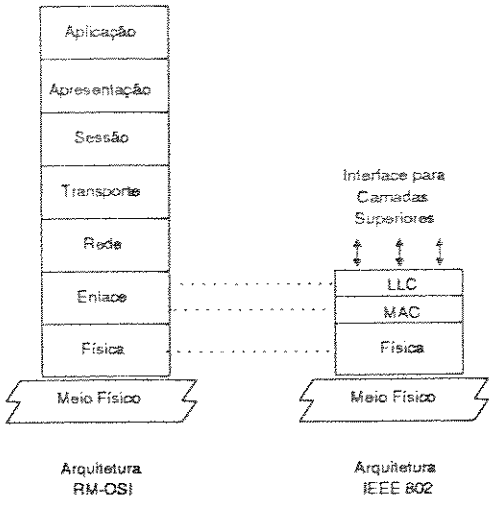

Figura 2.3: Proposta IEEE para Subdivisão em Camadas

A camada Física da arquitetura IEEE 802 provê os serviços básicos de transmissão e recepção de bits de maneira semelhante à camada de mesmo nome do Modelo de Referência OSI, definindo as características elétricas, mecânicas, funcionais e procedurais das conexões físicas.

A subcamada MAC ("Medium Access Control") fornece os serviços que disciplinam o compartilhamento de um meio de transmissão comum aos sistemas usuários da rede. A existência desta subcamada permite o desenvolvimento da subcamada superior com um certo grau de independência da camada Física, no que se refere à topologia e ao meio de transmissão.

A subcamada LLC ("Logical Link Control") encarrega-se de prover às camadas superiores um serviço confiável de transferência de sequências de bits entre os sistemas usuários da rede. Essa garantia da comunicação se faz através de mecanismos tais como: sequenciamento, controle de fluxo e recuperação de erros. A especificação do LLC prevê a existência de três tipos de serviços básicos. O mais simples deles fornece serviços de troca de unidades de dados sem o estabelecimento de conexão de enlace e sem confirmação, enquanto que o mais complexo é baseado no estabelecimento de conexões antes do início da troca de dados, incorporando os mecanismos citados acima. O terceiro tipo de serviço é um caso intermediário entre os dois primeiros, baseado em serviços de trocas de dados confirmadas, porém sem o estabelecimento de conexões de enlace.

A figura 2.4 mostra a família de padrões que compõem o Projeto 802 do IEEE e como eles se relacionam.

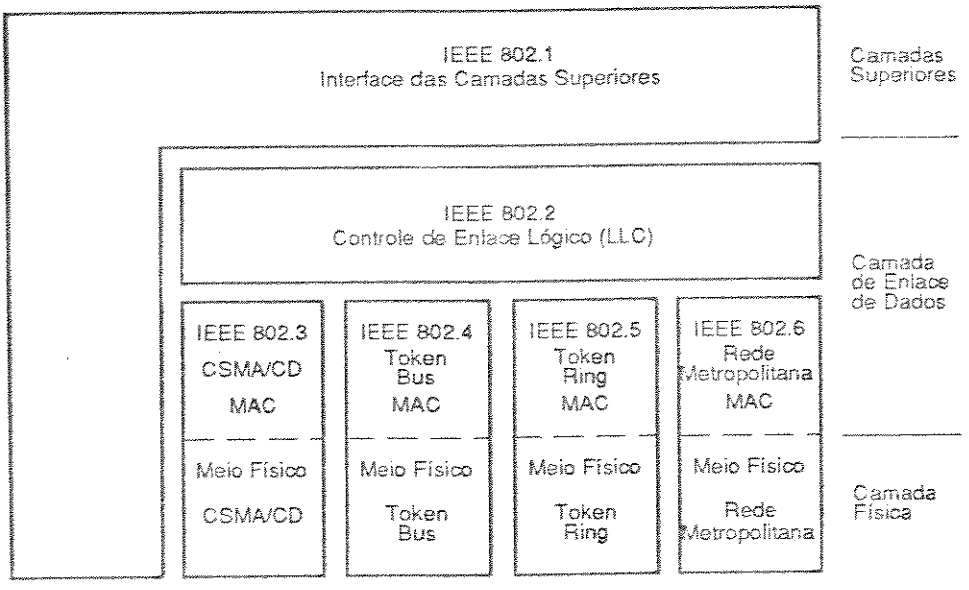

Figura 2.4: Organização dos Padrões do IEEE 802

O padrão IEEE 802.1 descreve o relacionamento entre os outros padrões e como estes se relacionam com o Modelo de Referência OSI, em maiores detalhes. Ele também explica a relação dos padrões com os protocolos das camadas superiores e discute, ainda, questões referentes ao endereçamento, interconexão e gerenciamento de redes.

O padrão IEEE 802.2 especifica um protocolo de controle de enlace lógico para ser utilizado com qualquer um dos outros padrões referentes aos métodos de controle de acesso e meio de transmissão associado.

O padrão IEEE 802.3 especifica um protocolo de acesso CSMA/CD num meio de transmissão configurado em barramento.

O padrão IEEE 802.4 especifica um barramento com método de controle de acesso ao meio "token-passing".

O padrão IEEE 802.5 especifica um anel com método de controle de acesso ao meio "token-passing".

O padrão IEEE 802.6 especifica um método de controle de acesso ao meio para uma subrede em áreas metropolitanas.

A figura 2.5 apresenta as opções de meios físicos e técnicas de codificação/modulação especificados nos padrões 802.3, 802.4 e 802.5 do IEEE.

Figura 2.5: Meios Físicos Especificados pelo IEEE 802

2.3 Redes Locais na Automação Industrial

O aumento de produtividade esperado pela automação, que ocorre atualmente em todos os níveis das plantas industriais, apoia-se diretamente no uso efetivo da tecnologia de redes de comunicação.

A rede local é um elemento pequeno, porém crucial, em toda a estratégia de manufatura integrada de uma empresa. A tecnologia de rede local torna possível a realização da comunicação entre todos os sistemas computacionais e dispositivos inteligentes associados com a manufatura.

O objetivo final da manufatura integrada é o de colocar todas as funções de uma organização de manufatura dentro de um único sistema orgânico, o qual pode responder rápida e adaptativamente às necessidades do mercado.

A coordenação de suas atividades externas em um propósito comum tem sido constantemente tentado pelas companhias de manufatura, porém seus sucessos no passado foram bastante limitados. O uso de computadores e, em particular, a conexão de computadores e dispositivos inteligentes em um mesmo sistema de comunicação, está alterando este cenário, tornando a idéia da manufatura integrada, o CIM ("Computer Integrated Manufacturing"), uma possibilidade real [HOLL86].

A utilização de uma rede local na área da automação da manufatura traz para o ambiente industrial as vantagens do sistema distribuído de controle, das quais pode-se salientar [BENN82]:

Custo

A colocação de microcomputadores próximos aos sensores e atuadores pode resultar em uma economia considerável em termos de cabos, se comparada à utilização de computadores centralizados. Além disso, o preço de compra dos vários microcomputadores para o sistema distribuído será, certamente, muito menor que o de um minicomputador que é empregado frequentemente em sistemas de controle centralizados.

Modularidade

O uso de módulos de processamento independentes, os quais se comunicam através do envio de mensagens, força a utilização de interfaces bem definidas, reduzindo a complexidade do software e, consequentemente, os custos de implementação. A expansão do sistema é bastante beneficiada com a modularidade, uma vez que novas funções podem ser incluídas pela simples adição de microcomputadores e não pelo aumento do poder de processamento ou troca por um computador maior.

Desempenho

O paralelismo resultante do uso de vários processadores permite que o sistema de controle distribuído possa ser utilizado com desempenho superior ao do sistema centralizado. Uma tarefa importante pode ser executada por um microcomputador dedicado, por exemplo, para satisfazer os requisitos de tempos mais restritos. E, ainda, microcomputadores podem

ser usados para pré-processamento de dados recolhidos, reduzindo o montante de informações transferidas para a rede de comunicação.

Confiabilidade

O sistema distribuído pode ser projetado de forma a continuar operando em caso de falhas, ainda que com seu desempenho degradado. A inclusão de esquemas de redundância no sistema de controle pode ser feita com menores custos.

Desenvolvimento e Manutenção

A inclusão de inteligência nos componentes do sistema de controle torna mais fácil o desenvolvimento do sistema por partes. Uma seção da planta industrial pode ser isolada para testes e manutenção.

Os principais problemas encontrados na implementação de sistemas distribuídos são:

- *complexidade do sistema de comunicação*: a complexidade e as dificuldades de implementação dos sistemas de comunicação são frequentemente subestimadas.
- *falta de padronização*: existem poucos padrões aceitos pelos diferentes fabricantes que permitiriam a conexão de seus equipamentos em um mesmo sistema de comunicação.

Mesmo com a adoção do Modelo de Referência OSI por parte de dois fabricantes de equipamentos, não se pode garantir que eles serão capazes de se comunicar entre si, uma vez que os protocolos usados em cada uma das camadas podem ser distintos.

A seleção dos protocolos que devem residir nas camadas deve ser feita em função das características desejadas para a rede local como, por exemplo, ambiente de utilização (indústria, escritório, campus, etc), alcance da rede (ligação com outras redes ou operação isolada) e confiabilidade na transferência de dados.

A *interoperabilidade* de equipamentos distintos requer, portanto, que cada um deles possua os mesmos protocolos em cada uma das suas camadas, selecionados de comum acordo entre os seus fabricantes. Entretanto, a seleção de um conjunto de protocolos para um uso específico não é uma tarefa fácil e não pode apenas expressar o interesse de alguns poucos fabricantes. A decisão de se adotar um certo conjunto de protocolos deve partir, principalmente, dos usuários dos equipamentos a serem interligados, os quais seriam os maiores beneficiários das vantagens decorrentes dessa interconexão.

Visando esses benefícios, empresas como a GM ("General Motors Co.") e a Boeing Co. americanas lideraram os esforços para se obter especificações de redes locais apropriadas para as suas aplicações mais prioritárias. O grupo de empresas encabeçado pela GM criou a especificação MAP ("Manufacturing Automation Protocol") [GEMO88], voltada para aplicações em automação industrial, enquanto que o grupo da Boeing surgiu com a especificação TOP ("Technical Office Protocol"), para automação de escritórios [MTUG88]. Também foi iniciado à cerca de três anos o desenvolvimento da especificação "Field-Bus", uma rede local para a interligação de dispositivos de automação primários, tais como, transdutores,

atuadores, "single-loops" e outros equipamentos de menor porte [IEC86] [MEND88] [ULME88].

Nos próximos itens, a especificação MAP será brevemente apresentada, uma vez que este trabalho baseou-se em uma de suas partes.

2.3.1 Especificação MAP

O final da década de 70 foi decisiva na história automobilística dos E.U.A., uma vez que o avanço dos japoneses nesse mercado fez com que os americanos percebecem a necessidade de se fazer algo para poder competir com a indústria japonesa. O caminho para o aumento dessa competividade passava, sem dúvida, pelo aperfeiçoamento da automação nas fábricas automobilísticas americanas [CROW85].

A GM americana adotou, nessa época, uma estratégia de automação em suas fábricas para encarar essa competição. Porém, ela se deu conta de que esse nível de automação exigiria o emprego de soluções oferecidas por um grande número de fornecedores de equipamentos e que o custo envolvido na solução dos problemas de incompatibilidade de comunicação entre esses equipamentos consumiria metade do orçamento destinado ao plano de automação [BABB87] [KAMI86].

Utilizando-se da sua grande potencialidade como usuário desses equipamentos, a GM criou em 1980 a *Força-Tarefa MAP*. O propósito do grupo era de desenvolver a política de trabalho na adoção de um sistema padronizado para todas as redes de comunicação de suas fábricas.

A Força-Tarefa MAP selecionou o Modelo de Referência OSI como a base para o MAP devido à aceitação mundial desse modelo. A adoção de padrões internacionais garantirá ao MAP reduzir riscos e incentivar, portanto, o empenho de fabricantes, os quais podem com isso atingir um mercado muito mais amplo que o dos E.U.A. para seus equipamentos. Também os grandes usuários, em todo o mundo, já perceberam os benefícios provenientes do uso de soluções de comunicação *não-proprietárias* baseadas no modelo OSI. Além disso, a seleção de padrões de comunicação internacionais tem uma outra vantagem muito importante, na medida que eles representam mais de 1000 homens-ano de esforços realizados por muitos dos maiores especialistas em comunicações no mundo.

Entretanto, o modelo OSI fornece uma estrutura ampla dentro da qual vários padrões distintos podem coexistir para satisfazer uma grande variedade de requisitos de comunicação, de forma que a conformidade a padrões do modelo OSI não garantem a interoperabilidade entre dois equipamentos distintos. Portanto, o trabalho da Força-Tarefa MAP foi o de definir, dentre os conjuntos de padrões mutualmente compatíveis, quais os convenientes ao ambiente industrial. As especificações MAP anteriores à versão 3.0 adotaram em algumas das suas camadas OSI as chamadas *especificações provisórias*, as quais eram tão próximas quanto possíveis dos padrões em fase de elaboração e foram necessárias para dar início às implementações e agilizar o desenvolvimento dos padrões internacionais. A GM, porém, comprometeu-se a trocar as especificações provisórias pelos padrões internacionais assim que eles se tornassem disponíveis.

A especificação MAP atual, versão 3.0, foi lançada em 1987 e deverá permanecer estável por 6 anos, sujeita apenas a erratas, encorajando o surgimento de produtos compatíveis e protegendo o investimento em hardware e software aplicativo dos usuários.

A seguir são apresentadas as arquiteturas MAP, MAP/EPA e Mini-MAP, como definidas na especificação MAP 3.0.

2.3.1.1 Arquitetura MAP

A figura 2.6 mostra como são configuradas as sete camadas do Modelo de Referência OSI na especificação MAP 3.0 [GEMO88].

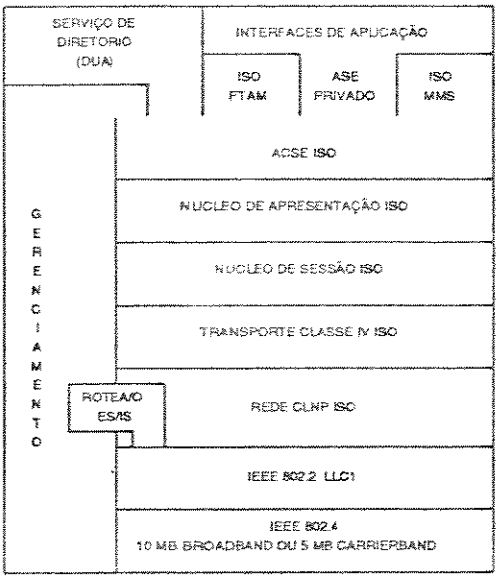

Figura 2.6: Arquitetura MAP Completa

A seguir apresenta-se um melhor detalhamento destas camadas.

Camada Física

Normas adotadas:

- IEEE 802.4 Draft J - Token-Passing Bus Access Method and Physical Layer. 1988.

O MAP adotou a topologia em barramento para meio físico por ele ter como vantagens: menor custo de fiação, de instalação e de expansão. Para a transmissão de dados neste meio físico adotou-se tanto a técnica "broadband" para a taxa de transmissão de 10 Mbit/seg, como a "carrierband" para 5 Mbit/seg, ambas definidas na norma IEEE 802.4. A primeira é indicada

para redes "backbone" e outras aplicações multi-canais, enquanto que a outra é indicada para as subredes e redes de canal-único.

Camada de Enlace

Normas adotadas:

- IEEE 802.4 Draft J - Token-Passing Bus Access Method and Physical Layer. 1988.
- IEEE 802.2 - Logical Link Control. 1987.

A escolha do protocolo "token-passing" de acesso ao meio definido pela norma IEEE 802.4 se deve a quatro razões principais [DETI88]: i) era a única técnica de acesso que o IEEE 802 especificava em um meio "broadband"; ii) é determinístico, ou seja, ele garante acesso à rede dentro de um período de tempo definido; iii) permite esquemas de prioridade de mensagens e serviço de *resposta imediata*; iv) muitos dos dispositivos programáveis passíveis de se conectar à rede MAP já eram voltados ao "token-bus". O gerenciamento do "token" é feito pela subcamada MAC que constitue a camada de Enlace segundo o projeto IEEE 802.

As outras responsabilidades da camada, as funções de Enlace, são executadas pela subcamada LLC. A norma IEEE 802.2 define três tipos de operações para o LLC: sem conexão e sem confirmação (Tipo 1), com conexão e com confirmação (Tipo 2) e sem conexão e com confirmação (Tipo 3). Dessas operações, a arquitetura MAP completa adotou a mais simples, o Tipo 1, uma vez que as funções de recuperação de erros, controle de fluxo e de sequenciamento são realizadas na camada de Transporte.

Camada de Rede

Normas adotadas:

- ISO 8348 - Network Service Definition.
- ISO 8348/AD1 - Addendum 1: Connectionless-mode Transmission.
- ISO 8348/AD2 - Addendum 2: Network Layer Addressing.
- ISO 8473+ AD1 - Protocol for Providing Connectionless-mode Network Service.
- ISO 8648 - Internal Organization of Network Layer.
- ISO/DIS 9542 - ES to IS Exchange Protocol.

Para esta camada o MAP adotou os serviços sem conexão CLNS ("ConnectionLess-mode Network Service") implementáveis pela norma ISO 8473+ AD1. O MAP seleciona protocolos e serviços para cada um dos módulos definidos na organização interna da camada de Rede pela norma ISO 8648: o protocolo de convergência independente da subrede (SNICP), o protocolo de convergência dependente da subrede (SNDCP) e o protocolo de acesso à subrede (SNAcP).

A especificação do modelo de roteamento e do formato do endereçamento no MAP é baseada nas normas ISO 8348/AD2 e ISO/DIS 9542.

Camada de Transporte

Normas adotadas:

- ISO 8072 - Tranport Service Definition. 1984.
- ISO 8073 - Connection-Oriented Transport Protocol Specification. 1984.

A especificação MAP adotou , para esta camada, o conjunto de serviços mais complexo definido no padrão ISO 8072: o da Classe 4. Esta classe fornece as funções básicas de conexão/desconexão, transferência de dados com segmentação e remontagem, multiplexação de conexões de Transporte (opcional), controle de fluxo, sequenciamento de dados, detecção e recuperação de erros e envio de dados acelerados. O uso de "checksum" é uma opção negociável na Classe 4.

Camada de Sessão

Normas adotadas:

- ISO 8326 - Basic Connection-Oriented Session Service Definition. 1987.
- ISO 8327 - Basic Connection-Oriented Protocol Specification. 1987.
- ISO 8326/DAD2 - Basic Oriented Session Service Definition to Incorporate Unlimited User Data. 1987.
- ISO 8327/DAD2 - Basic Oriented Session Protocol Specification to Incorporate Unlimited User Data. 1987.

Os serviços disponíveis nas normas adotadas para esta camada são organizados em *unidades funcionais* as quais são negociadas na fase de estabelecimento de conexão. Dessas unidades funcionais o MAP selecionou três para seu serviço de Sessão:

- a *Unidade Funcional do Núcleo* executa os serviços básicos de estabelecimento e término de conexões de Sessão e os de transferência de dados normais. Esta unidade não é negociável;
- a *Unidade Funcional Duplex* realiza os serviços de transferência de dados do tipo "full-duplex", isto é, nos dois sentidos simultâneamente;
- a *Unidade Funcional de Ressincronização* permite que o usuário da camada de Sessão coloque a conexão de Sessão em pontos intermediários de sincronização no fluxo de dados, possibilitando retornar a pontos anteriores em caso de erros. Esta unidade foi incluída para uso futuro, não sendo utilizada atualmente.

Camada de Apresentação

Normas adotadas:

- ISO 8822 - Connection-Oriented Presentation Service Definition. 1988.
- ISO 8823 - Connection-Oriented Presentation Protocol Definition. 1988.
- ISO 8824 - Specification of Abstract Syntax Notation One (ASN.1). 1987.
- ISO 8825 - Specification of Basic Encoding Rules for ASN.1. 1987.

Das normas ISO 8822 e ISO 8823 a arquitetura MAP adota a Unidade Funcional do Núcleo de Apresentação para executar os serviços básicos de estabelecimento e término de conexões de Apresentação e os de transferência de dados normais. São também adotadas as sintaxes abstrata e de transferência para serem utilizadas com o FTAM, MMS, Serviços de Diretório e Gerenciamento da Rede.

Camada de Aplicação

Os programas aplicativos ao desejarem fazer uma *associação* com outra entidade na rede ativam os serviços do ACSE ("Association Control Service Element"), que negocia os parâmetros comuns de comunicação. Junto na associação encontra-se um ASE ("Application Service Element") o qual fornece o tipo de serviço requerido pelo programa aplicativo. Os quatro ASEs especificados pelo MAP são: FTAM, MMS, Serviços de Diretório e Gerenciamento da Rede.

- ACSE

Normas adotadas:

- ISO 8649/2 - Service Definition for Common Application Service Elements - Part 2. Association Control. 1987.
- ISO 8650/2 - Protocol Specification for Common Application Service Elements - Part 2. Association Control. 1987.

Este elemento fornece os serviços para o estabelecimento e término das associações de Aplicação.

- FTAM ('File Transfer, Access and Management")

Normas adotadas:

- ISO 8571/1 - FTAM Part 1: General Introduction. 1987.
- ISO 8571/2 - FTAM Part 2: The Virtual Filestore. 1987.
- ISO 8571/3 - FTAM Part 3: The File Service Definition. 1987.
- ISO 8571/4 - FTAM Part 4: The File Protocol Definition. 1987.

Este ASE fornece um conjunto de serviços usados na transferência de informações entre processos de aplicação e armazenadores de arquivos remotos. O FTAM permite operar com arquivos binários e de texto, criar arquivos, apagar o conteúdo de um arquivo, localizar "records" específicos e ler e escrever "records" de um arquivo.

- MMS ("Manufacturing Message Specification")

Normas adotadas:

- ISO/DIS 9506/1 - MMS - Service Definition. 1987.
- ISO/DIS 9506/2 - MMS - Protocol Specification. 1987.

O MMS é um serviço de mensagem genérico, de forma que seu vocabulário foi definido para cobrir os tipos de interações comuns para a maior gama possível de dispositivos

programáveis na fábrica. Os serviços de mensagem para dispositivos de fábrica que necessitam de informações específicas, são definidos em quatro "companion standards" cobrindo: *controladores numéricos* (especificação EIA), *controladores programáveis* (especificação NEMA), *controladores de robôs* (especificação RIA) e *sistemas de controle de processos* (especificação ISA).

- Serviços de Diretório

Normas adotadas:

- ISO/DIS 9594/1-8. 1987.
- ISO/DIS 9072/1 - Remote Operations: Model Notation and Service Definition, Version 5. 1987.
- ISO/DIS 9072/2 - Remote Operations: Protocol Specification, Version 5. 1987.

Estes serviços são baseados em uma Base de Informações de Diretório, a qual armazena o nome de cada objeto acessível na rede juntamente com o endereço pelo qual ele pode ser alcançado.

- Gerenciamento da Rede

Normas adotadas:

- ISO/DIS 7498/4 - OSI Management Framework.
- ISO/DP 9595/1 - Management Information Service Definition - Part 1. Overview. 1986.
- ISO/DP 9595/2 - Management Information Service Definition - Part 2. Common Management Information Service Definition. 1986.
- ISO/DP 9596/1 - Management Information Protocol Specification - Part 1. Overview. 1986.
- ISO/DP 9596/2 - Management Information Protocol Specification - Part 2. Common Management Information Protocol. 1986.
- ISO/DIS 9072/1 - Remote Operations: Model Notation and Service Definition, Version 5. 1987.
- ISO/DIS 9072/2 - Remote Operations: Protocol Specification, Version 5. 1987.
- Draft IEEE Standard 802.1: Part B - Systems Management Protocol, Revision L. 1896.
- Draft IEEE Standard 802.1: Part B - Systems Load Protocol, Revision A. 1986.

O MAP utiliza os serviços do Gerenciamento da Rede para desempenhar três papéis principais: i) coletar informações sobre a utilização da rede pelos dispositivos nela conectados; ii) garantir a operação correta da rede; iii) fornecer relatórios.

- Interface de Aplicação

Para garantir uma fácil portabilidade de programas aplicativos que interagem com a rede, entre computadores que usam a mesma linguagem de programação, encontra-se em desenvolvimento a Interface de Aplicação. Além de cobrir o Gerenciamento de Conexão e as Funções de Suporte, ela abrange as comunicações via FTAM, MMS e proprietária.

Atualmente a Interface de Aplicação encontra-se baseada na linguagem de programação C, independente do sistema operacional usado. A especificação MAP inclui a Interface de Aplicação no seu apêndice 7.

2.3.1.2 Arquitetura Mini-MAP

Simulações realizadas com a arquiteruta MAP mostraram que tempos de resposta em torno de 20 a 30 mseg podem ser atingidos na ausência de erros e, talvez, esses tempos sejam duplicados na ocorrência de erros. A GM espera poder conviver com tempos de resposta da ordem de 50 mseg em suas aplicações, utilizando arquiteturas MAP [HOLL86].

Entretanto, à medida que a especificação MAP progrediu, usuários da área de controle de processos sentiram a necessidade de um subconjunto dos serviços MAP, os quais sacrificariam alguma funcionalidade em troca de tempos de respostas mais rápidos. Estas adições à especificação MAP foram encaminhadas através de um grupo de trabalho do Subcomitê de Dispositivos Programáveis, que produziram duas arquiteturas alternativas em resposta a essas preocupações.

A primeira dessas arquiteturas, a qual atinge o requisito de velocidade abandonando a compatibilidade com o modelo OSI, é conhecida como Mini-MAP. A incompatibilidade dessa arquitetura com o modelo OSI ocorre pela ausência das camadas 3, 4, 5 e 6, como mostra a figura 2.7.

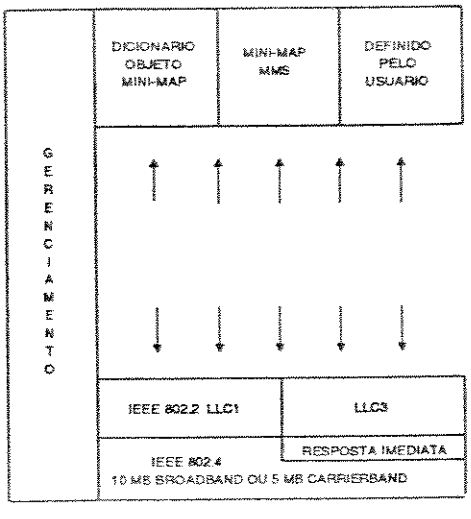

Figura 2.7: Arquitetura Mini-MAP

A arquitetura Mini-MAP introduz uma alteração na subcamada LLC pela adoção da operação Tipo 3 com serviços sem conexão e com confirmação, usando o conjunto de opções existentes na especificação PROWAY da ISA. A operação Tipo 3 do LLC é assunto deste trabalho e será descrito no capítulo 3.

A camada de Aplicação utiliza o protocolo MMS definido pelas normas ISO/DIS 9506/1 e /2.

A camada de Enlace utiliza na subcamada LLC a operação Classe 3 definida no padrão IEEE 802.2, oferecendo os serviços disponíveis tanto na operação Tipo 1 como na operação Tipo 3. A subcamada MAC utiliza o padrão IEEE 802.4 (protocolo "token-passing") com as opções adequadas às aplicações de tempo-real.

A camada Física pode utilizar qualquer meio físico padrão IEEE 802.4 permitido para os nós MAP, incluindo o "broadband" e o "carrierband". Entretanto, é esperado que a maioria das implementações de redes Mini-MAP (e MAP/EPA) irá usar a tecnologia "carrierband" nesta camada.

2.3.1.3 Arquitetura MAP/EPA

A segunda arquitetura proposta pelo grupo de trabalho do Subcomitê de Dispositivos Programáveis é a denominada MAP/EPA ("Enhanced Performance Architecture"). Esta arquitetura é caracterizada pelas presenças tanto da arquitetura MAP como da arquitetura Mini-MAP no mesmo nó de comunicação, como mostra a figura 2.8.

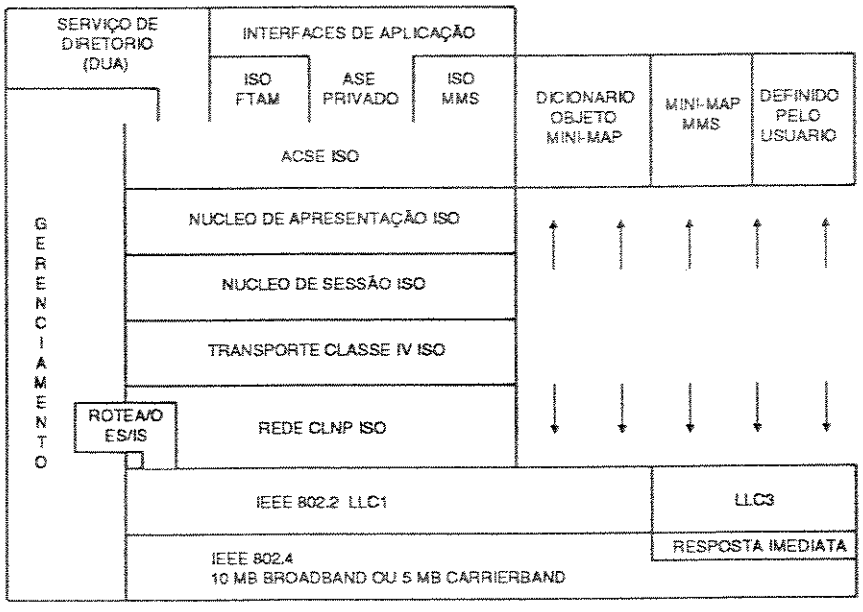

Figura 2.8: Arquitetura MAP/EPA

A principal razão para a existência da arquitetura MAP/EPA é a de oferecer aos nós MAP uma forma de obter tempos de respostas mais rápidos ao se utilizar redes locais em aplicações que possuem restrições de tempo críticas.

Deve-se notar que as camadas de Enlace e Física na arquitetura MAP/EPA são as mesmas adotadas na arquitetura Mini-MAP. Isso permite que nós MAP/EPA comuniquem-se diretamente com nós Mini-MAP em um mesmo segmento de rede local, ou com nós MAP em uma rede "backbone" através do uso de um nó *ponte*, como exemplifica a figura 2.9.

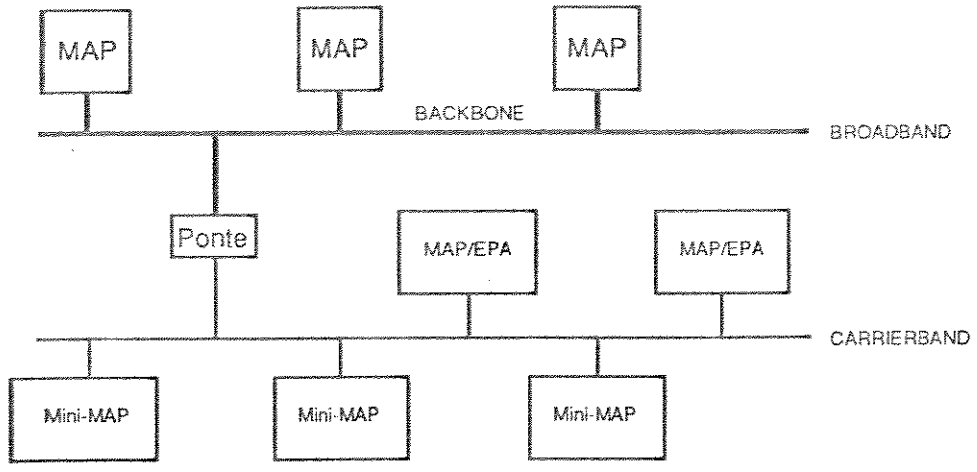

Figura 2.9: Exemplo de Interligação das Arquiteturas

Uma utilização da estrutura da figura acima seria, por exemplo, supor que um dos nós MAP/EPA é um controlador de célula de manufatura, os nós Mini-MAP são os controladores programáveis dessa célula e os nós MAP são os computadores de gestão de produção da fábrica. O controlador de célula recebe uma programação de produção de um dos computadores de gestão utilizando-se dos serviços de transferência de arquivos do protocolo FTAM, suportado pela metade MAP da sua arquitetura MAP/EPA. Uma vez com o programa de produção carregado, o controlador de célula interage com os seus controladores programáveis utilizando-se do protocolo MMS da sua metade Mini-MAP.

2.3.2 Generalidades sobre Produtos MAP

Este item visa dar um apanhado geral da situação atual do desenvolvimento de produtos MAP no mercado internacional, com destaque especial para as placas de comunicação comercializadas por três dos mais importantes fabricantes de produtos MAP nos E.U.A.: a Motorola, a Computrol e a Concord.

O objetivo desta discussão não é o de ser um levantamento completo dos produtos MAP existentes no mercado internacional e sim, o de apresentar alguns produtos, destacando

algumas de suas características principais para servirem como parâmetros de comparação em uma avaliação posterior da implementação de que trata este trabalho de dissertação.

2.3.2.1 Características de Hardware

A tabela 2.1 fornece as características principais presentes no hardware das placas de rede local fornecidas pelos fabricantes citados acima.

CARACTERISTICA / FABRICANTE	PADRÃO DO BARRAMENTO	CHIP CONTROLADOR DE REDE	CHIP MICRO PROCESSADOR	QUANTIDADE MEMORIA RAM
COMPUTROL	PC XT/AT MULTI-BUS O-BUS VME	MC 68824	MC 68020 (12,5 Mhz)	1 Mbit
CONCORD	PC XT/AT PS/2	MC 68824	INTEL 80C186 (12,5 Mhz)	1 Mbit
MOTOROLA	VME	MC 68824	MC 68020 (12,5 Mhz)	640 Kbit

Tabela 2.1: Características do Hardware das Placas MAP

Todas as placas suportam as sete camadas da especificação MAP 3.0, daí a necessidade de se ter a potência computacional oferecida pelos microprocessadores de 16/32 bits, além da capacidade de endereçamento de memória permitida por eles.

A interface entre a placa de rede local e o barramento do sistema hospedeiro é feita, normalmente, através da utilização de endereços de entrada/saída (E/S), seja para a troca de dados como para controle e "status", a menos da placa Motorola que possui uma área de RAM ("Random Access Memory") compartilhada entre o barramento VME, o microprocessador e o controlador de rede.

Todos os fabricantes oferecem modems dos tipos "broadband" e "carrierband" separados da placa de rede. À exceção da Concord, os outros fabricantes possuem os modems confeccionados em uma placa gêmea, que ocupa um "slot" vizinho à placa de rede no barramento do sistema hospedeiro e dele retira sua alimentação; a conexão da placa modem à placa de rede é feita via conectores apropriados compatíveis com o padrão IEEE 802.4. Desejando-se evitar o uso de um "slot" extra do barramento, a Concord oferece modems "broadband" e "carrierband" externos, com alimentação independente e caixa própria.

2.3.2.2 Características de Software

Os fabricantes desses produtos MAP fornecem "drivers" de software para uma ampla gama de sistemas operacionais dos sistemas hospedeiros, podendo se destacar: MS-DOS, UNIX, OS/2, VMS, RSX-11+ e VERSAdos. Esses "drivers" ficam residentes no hospedeiro e são necessários para a operação adequada de uma certa placa de rede de um fabricante no sistema operacional usado pelo hospedeiro.

Era de se esperar que as placas de rede abrigassem integralmente as sete camadas do MAP e que o sistema hospedeiro somente possuísse as interfaces dos serviços de aplicação (FTAM, MMS e ACSE) e os "drivers". Porém, não é isso que ocorre na realidade. Como exemplo, pode-se citar o produto MMS-EASE desenvolvido pela firma americana SISCO ("Systems Integration Specialists Company"), o qual é utilizado nas placas de rede dos três fabricantes aquí citados; a figura 2.10 mostra como esse software é alocado no sistema que constitue um nó de rede.

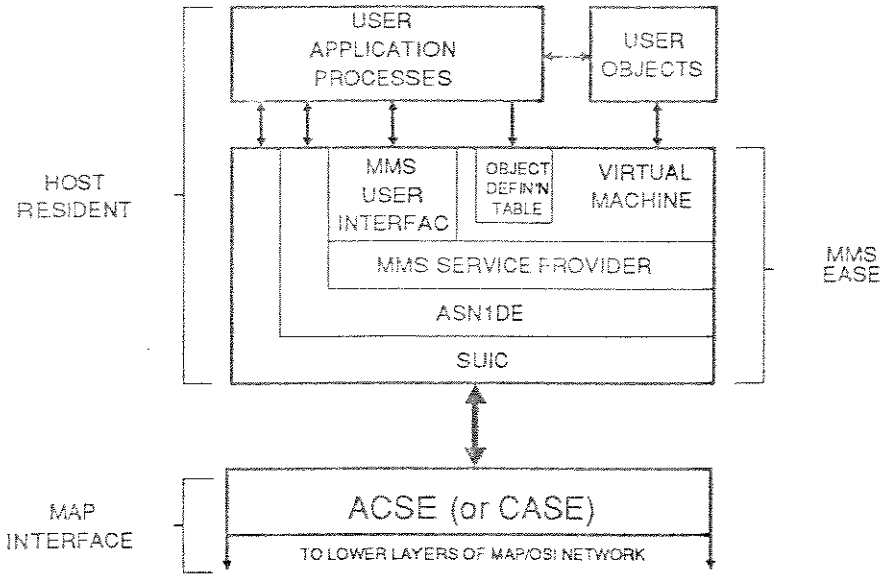

Figura 2.10: Localização do MMS-EASE no Nó de Rede

Todos os três fabricantes adotam a mesma forma de colocar as camadas de protocolo na memória interna de suas placas de rede, ou seja, as camadas são carregadas em memória RAM, a partir do sistema hospedeiro, após ter sido completada a fase de configuração do nó de rede. Esta fase compreende a definição dos parâmetros de operação do nó de rede (como por exemplo: contadores e temporizadores internos), assim como o registro dos processos de aplicação locais e remotos, com os quais o nó irá se comunicar.

A vantagem de se carregar as camadas de protocolo em RAM, ao invés de tê-las residentes em EPROM ("Erasable Programmable Read Only Memory") na placa de rede, é a de que a maioria dos parâmetros de operação do nó podem ser definidos "off-line", na configuração, sem a necessidade de se implementar programas responsáveis pela iniciação e alteração desses parâmetros. Nesta classe de programas se encontram muitos daqueles que tratam as primitivas das interfaces entre as camadas de protocolo e a Gerência da Estação (ver item 4.2.2). Esse procedimento de carregar as camadas de protocolo permite uma sensível simplificação do software no que se refere ao tamanho dos programas gerados, entretanto, ele tem como inconveniente a necessidade de se repetir todo o processo de carregamento se um dos parâmetros de operação precisar ser alterado, quando o nó de rede já se encontra em operação.

Comum, também, a todos esses fabricantes é a utilização de um ambiente multi-tarefas nas placas de rede, com a adoção de sistemas operacionais de tempo-real. Na placa de rede da Computrol, por exemplo, cada camada é implementada como uma tarefa independente para o sistema operacional. A Motorola utiliza na sua placa de rede o sistema operacional de tempo-real VRTX da Hunter & Ready Inc.

Quanto à linguagem de programação usada na implementação das camadas de protocolo, todos os fabricantes fazem uso da linguagem C. Se ela é empregada em todo o software não está especificado, mas as interfaces fornecidas ao usuário, pelo menos, são escritas em C.

A placa Concord "Series 1210" para PC suporta até 64 associações de aplicação simultâneas.

Os fabricantes Motorola e Computrol já anunciaram produtos Mini-MAP utilizando algumas das placas de rede que suportam o MAP completo.

Capítulo 3

Subcamada LLC

Três tipos de operação da subcamada LLC são definidos de maneira a melhor atender as diversas necessidades de comunicação entre sistemas, independentemente do método de acesso ao meio usado na rede local.

Operação Tipo 1

Oferece serviços sem conexão e sem confirmação, caracterizando-se por um protocolo de complexidade mínima que pode ser útil no caso em que as camadas superiores fornecem serviços de recuperação de erros e sequenciamento de mensagens. Ainda, este serviço pode ser útil em aplicações onde a garantia no envio de dados não é essencial.

Operação Tipo 2

Oferece serviços baseados em conexões de Enlace de Dados, incluindo mecanismos de sequenciamento, controle de fluxo e recuperação de erros, comparáveis aos procedimentos de controle de Enlace de Dados já padronizados, tais como o HDLC (ISO e CCITT) e ADCCP (ANSI).

Operação Tipo 3

Oferece serviços sem conexão e com confirmação, os quais permitem que uma estação tanto envie como requisite dados de uma estação remota ao mesmo tempo. Mesmo sendo serviços não confirmados, o sequenciamento do envio de mensagens é garantido pela estação que inicia a transferência.

O Mini-MAP utiliza a operação Tipo 3 no LLC por este ser o protocolo mais simples para se ter uma transferência confiável de PDUs. A inexistência de mecanismos de recuperação de erros na camada de Aplicação não permite que a operação Tipo 1 seja utilizada, enquanto que a complexidade da operação Tipo 2 está acima das necessidades de uma rede local deste tipo, no que se refere aos serviços oferecidos, além de adicionar um

"overhead" computacional indesejado em uma rede que visa atender aplicações em tempo-real.

3.1 Características do Protocolo LLC Tipo 3

Os serviços sem conexão e com confirmação oferecidos pelo LLC Tipo 3 fornecem os meios pelos quais entidades usuárias da camada de Enlace de Dados podem trocar unidades de dados que são confirmadas à nível da subcamada LLC sem o estabelecimento de uma conexão de Enlace.

Esses serviços permitem que uma entidade usuária da camada de Enlace em uma estação possa:

- enviar unidades de dados para outra estação;
- requisitar, de uma outra estação, unidades de dados previamente preparadas;
- enviar unidades de dados ao LLC local para o acesso posterior por parte de uma outra estação, na forma descrita no item anterior.

A figura 3.1(a) ilustra o diagrama de sequência no tempo para os serviços citados nos dois primeiros itens acima, enquanto que a figura 3.1(b) ilustra o mesmo diagrama para o serviço descrito no último item.

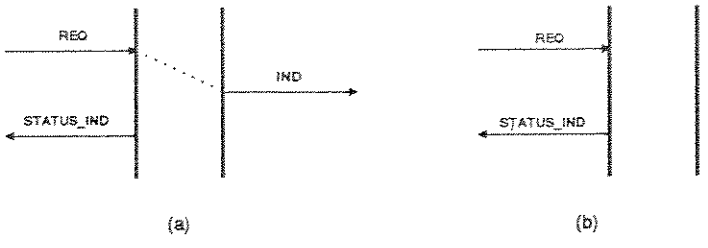

Figura 3.1: Diagramas de Sequência dos Serviços Tipo 3

Ao receber uma primitiva do tipo REQUEST para enviar uma unidade de dados, por exemplo, o LLC local gera uma *PDU de Comando* que é enviada ao LLC remoto através da rede local. O formato da PDU de Comando LLC é mostrado na figura 3.2 [IEEE84], onde o campo DSAP especifica o SAP do LLC destino e o campo SSAP especifica o SAP do LLC fonte.

Figura 3.2: PDU de Comando do LLC Tipo 3

Quando a PDU de Comando é recebida pelo LLC remoto, a unidade de dados é passada ao usuário da camada de Enlace na forma de uma primitiva do tipo INDICATION. Na sequência, o LLC prepara uma *PDU de Resposta* e a envia ao LLC local. O formato da PDU de Resposta LLC é mostrado na figura 3.3 [ISO86].

Figura 3.3: PDU de Resposta do LLC Tipo 3

Ao receber a PDU de Resposta, o LLC local passa o campo de STATUS e a unidade de dados (se houver) para o usuário da camada de Enlace, na forma de uma primitiva do tipo STATUS_INDICATION, a qual é a confirmação que encerra um serviço de troca de dados do Tipo 3.

O campo de STATUS se subdivide nos subcampos CCCC e RRRR, os quais trazem informações sobre a transferência de uma unidade de dados, a saber:

- *subcampo CCCC*: os códigos nele retornados indicam o sucesso ou falha na passagem de informações contidas na PDU de Comando, do LLC que a gerou para o LLC destino;
- *subcampo RRRR*: os códigos nele retornados indicam o sucesso ou falha na passagem de informações na PDU de Resposta, por parte do LLC que está respondendo.

As tabelas 3.1 e 3.2 apresentam os códigos de erro referentes aos subcampos CCCC e RRRR, respectivamente.

CCCC	MNEMONICO	CATEGORIA	DESCRIÇÃO
0000	OK	Sucesso	Comando aceito
1000	RS	Erro Perm	Serviço não implementado ou desativado
1010	UE	Erro Perm	Erro da interface com o usuário do LLC
0110	PE	Erro Perm	Erro de protocolo
1110	IP	Erro Perm	Erro permanente dependente da implementação
1001	UN	Erro Temp	Recursos não disponíveis temporáriamente
1111	IT	Erro Temp	Erro temporário dependente da implementação

Tabela 3.1: Códigos de Erro do Subcampo CCCC

RRRR	MNEMONICO	CATEGORIA	DESCRIÇÃO
0000	OK	Sucesso	LSDU resposta presente
1000	RS	Erro Perm	Serviço não implementado ou desativado
1100	NE	Erro Perm	LSDU resposta nunca enviada
0010	NR	Sucesso	LSDU resposta não requisitada
1010	UE	Erro Perm	Erro da interface do usuário do LLC
1110	IP	Erro Perm	Erro permanente dependente da implementação
1001	UN	Erro Temp	Recursos não disponíveis temporáriamente
1111	IT	Erro Temp	Erro temporário dependente da implementação

Tabela 3.2: Códigos de Erro do Subcampo RRRR

O envio de PDUs sucessivas no protocolo LLC Tipo 3 segue um mecanismo de sequenciamento binário, que permite ao LLC diferenciar uma PDU de Comando recebida, de uma cópia recebida anteriormente. O número da sequência de uma PDU é embutido no seu campo CONTROLE e, em função dele, as PDUs LLC do Tipo 3 são denominadas: *PDUs de Comando AC0/AC1* ou *PDUs de Resposta AC0/AC1*, dependendo do valor do número de sequência binário no momento em que a PDU é criada.

3.2 Serviços Oferecidos pelo LLC Tipo 3

Neste item são descritas as primitivas associadas aos serviços disponíveis na operação Tipo 3 da subcamada LLC. Cada primitiva possui parâmetros próprios, que são descritos a seguir:

- *source_address*: especifica os SAPs locais envolvidos na transmissão de unidades de dados. Este parâmetro é formado pela concatenação lógica dos endereços do MAC (MSAP local ou SA) e do LLC (LSAP local ou SSAP);
- *destination_address*: especifica os SAPs remotos envolvidos na transmissão de unidades de dados. Este parâmetro é formado pela concatenação lógica dos endereços do MAC (MSAP remoto ou DA) e do LLC (LSAP remoto ou DSAP);
- *data*: especifica a SDU ("Service Data Unit") de Enlace recebida pela entidade da subcamada LLC;
- *priority*: especifica a prioridade associada à transmissão de unidades de dados;
- *service_class* especifica se um recurso de confirmação, na subcamada MAC, foi ou será utilizado na transferência da unidade de dados;
- *status*: indica o sucesso ou falha na execução de um pedido de transmissão de uma unidade de dados do Tipo 3, feita anteriormente.

As primitivas são agrupadas de acordo com o tipo de serviço que elas fornecem ao usuário da subcamada LLC.

Serviço de transmissão de unidades de dados

```
DL_DATA_ACK request (
                        source_address,
                        destination_address,
                        data,
                        priority,
                        service_class
                        )

DL_DATA_ACK indication (
                        source_address,
                        destination_address,
                        data,
                        priority,
                        service_class
                        )

DL_DATA_ACK_STATUS indication (
                        source_address,
                        destination_address,
                        priority,
                        service_class,
                        status
                        )
```

Este serviço é usado apenas para o envio de unidades de dados entre dois usuários da camada de Enlace em estações distintas. Neste caso, a primitiva de confirmação ("DL_DATA_ACK_STATUS indication") recebida pelo usuário que gerou o "request" nunca transportará dados do outro usuário.

Serviço de troca de unidades de dados

```
DL_REPLY request (
                        source_address,
                        destination_address,
                        data,
                        priority,
                        service_class
                        )
```

DL_REPLY indication (
source_address,
destination_address,
data,
priority,
service_class
)

DL_REPLY_STATUS indication (
source_address,
destination_address,
data,
priority,
service_class,
status
)

Este serviço é utilizado essencialmente para requisitar unidades de dados de uma estação remota, entretanto o envio de unidades na primitiva de "request" é também permitida, ocorrendo assim uma troca de dados entre estações. A unidade de dados requisitada é retornada para a estação requisitante na primitiva "DL_DATA_ACK_STATUS indication", juntamente com informações de "status", sempre presentes nesta primitiva.

Serviço de preparação de unidades de dados

DL_REPLY_UPDATE request (
source_address,
data
)

DL_REPLY_UPDATE_STATUS indication (
source_address,
status
)

Este serviço é utilizado pelo usuário da camada de Enlace para armazenar uma unidade de dados em um "Reply-Buffer" localizado no LLC local, para que ela seja enviada, posteriormente, a uma outra estação quando esta fizer a solicitação, utilizando-se do serviço de troca de unidades de dados.

3.3 Operação do LLC Tipo 3

Em operação normal, cada PDU de Comando LLC deve receber uma PDU de Resposta, como confirmação. O LLC fonte pode retransmitir uma PDU de Comando com a finalidade de recuperação de erros, porém não lhe é permitido enviar uma nova PDU através de um certo LSAP local e a uma dada prioridade MAC, para um LLC em uma dada estação (endereço DA), sem que ele tenha recebido uma confirmação de uma PDU enviada anteriormente, envolvendo os mesmos endereços e prioridade.

Assim, para modelar essas restrições, o protocolo LLC do Tipo 3 subdivide suas funções entre dois componentes: o *Componente de Transmissão* e o *Componente de Recepção*.

Componente de Recepção (CR)

É responsável pelo processamento de PDUs de Comando do Tipo 3 recebidos da subcamada MAC, e pelo envio de PDUs de Resposta do Tipo 3 para os originadores dos comandos. Em uma dada estação existe, potencialmente, um Componente de Recepção do Tipo 3 distinto para cada possível combinação dos endereços remotos SA e SSAP, e da prioridade MAC. Entretanto, em uma certa aplicação somente existirão os Componentes de Recepção para os quais PDUs de Comando Tipo 3 são realmente recebidas.

Cada Componente de Recepção possui suas próprias variáveis de estado V(RI) e V(RB), e seu temporizador de duração das variáveis de recepção. A variável V(RI) é utilizada no teste de duplicidade das PDUs de Comando recebidas, a variável V(RB) é utilizada no teste do "status" de uma recepção anterior e o temporizador define, basicamente, o tempo de existência do Componente de Recepção.

A tabela 3.3 apresenta a tabela de transição de estados para o Componente de Recepção, definida em [ISO86].

ESTADO	EVENTO	AÇÃO	PROX_EST
READY	REPLY_UPDATE_REQUEST	SAVE:= GIVEN LSDU REPLY_UPDATE_STATUS_INDICATE	READY
	RECEIVE ACn CMD(SOC=V(RI), P=0, INFO<>NULL) and RECEIVE_STATUS()=OK	SEND_ACn_RSP(SQR=1-SOC, F=0, C=OK, R=NR, LSDU=NULL) DATA_ACK_INDICATE V(RI):=1-SOC V(RB):=OK START_RCV_LIFE_TIMER	READY
	RECEIVE ACn CMD(SOC=V(RI), P=0, INFO=NULL) and RECEIVE_STATUS()=OK	SEND_ACn_RSP(SQR=1-SOC, F=0, C=OK, R=NR, LSDU=NULL) V(RI):=1-SOC V(RB):=OK START_RCV_LIFE_TIMER	READY
	RECEIVE ACn CMD(SOC=V(RI), P=0) and RECEIVE_STATUS()<>OK	SEND_ACn_RSP(SQR=1-SOC, F=0, C=RECEIVE_STATUS(), R=NR, LSDU=NULL) V(RI):=1-SOC V(RB):=RECEIVE_STATUS() START_RCV_LIFE_TIMER	READY
	RECEIVE ACn CMD(SOC=V(RI), P=1) and RECEIVE_STATUS()=OK and ACCESS()=OK	SEND_ACn_RSP(SQR=1-SOC, F=1, C=OK, R=OK, LSDU=SAVE) REPLY_INDICATE(LSDU=INFO) V(RI):=1-SOC V(RB):=OK START_RCV_LIFE_TIMER	READY
	RECEIVE ACn CMD(SOC=V(RI), P=1) and RECEIVE_STATUS()<>OK and ACCESS()=OK	SEND_ACn_RSP(SQR=1-SOC, F=1, C=RECEIVE_STATUS(), R=OK, LSDU=SAVE) REPLY_INDICATE(LSDU=NULL) V(RI):=1-SOC V(RB):=RECEIVE_STATUS() START_RCV_LIFE_TIMER	READY
	RECEIVE ACn CMD(SOC=V(RI), P=1, INFO<>NULL) and RECEIVE_STATUS()=OK and ACCESS()<>OK	SEND_ACn_RSP(SQR=1-SOC, F=1, C=OK, R=ACCESS(), LSDU=NULL) DATA_ACK_INDICATE V(RI):=1-SOC V(RB):=OK START_RCV_LIFE_TIMER	READY
	RECEIVE ACn CMD(SOC=V(RI), P=1, INFO=NULL) and RECEIVE_STATUS()=OK and ACCESS()<>OK	SEND_ACn_RSP(SQR=1-SOC, F=1, C=OK, R=ACCESS(), LSDU=NULL) V(RI):=1-SOC V(RB):=OK START_RCV_LIFE_TIMER	READY

Tabela 3.3: Tabela de Transição de Estados do CR

ESTADO	EVENTO	AÇÃO	PROX_EST
READY	RECEIVE ACn CMD(SQC=V(RI), P=1) and RECEIVE_STATUS()<>OK and ACCESS()<>OK	SEND_ACn_RSP(SQR=1-SQC, F=1, C=RECEIVE_STATUS(), R=ACCESS(), LSDU=NULL) V(RI):=1-SQC V(RB):=RECEIVE_STATUS() START_RCV_LIFE_TIMER	READY
	RECEIVE ACn CMD(SQC<>V(RI), P=0)	SEND_ACn_RSP(SQR=1-SQC, F=0, C=V(RB), R=NR, LSDU=NULL)	READY
	RECEIVE ACn CMD(SQC<>V(RI), P=1) and ACCESS()=OK	SEND_ACn_RSP(SQR=1-SQC, F=1, C=V(RB), R=OK, LSDU=SAVE) REPLY_INDICATE(LSDU=NULL)	READY
	RECEIVE ACn CMD(SQC<>V(RI), P=0) and ACCESS()<>OK	SEND_ACn_RSP(SQR=1-SQC, F=1, C=V(RB), R=ACCESS, LSDU=NULL)	READY
	RCV_LIFE_TIMER_EXPIRED	DESTROY_V(RI)	READY

Tabela 3.3: Tabela de Transição de Estados do CR (cont.)

Componente de Transmissão (CT)

É responsável pelo envio de PDUs de Comando do Tipo 3 para um LLC remoto. Ele também recebe PDUs de Resposta e re-envia PDUs de Comando, caso suas respostas não sejam recebidas dentro de um certo período de tempo. O protocolo LLC Tipo 3 permite apenas uma PDU de Comando pendente (ainda não confirmada) para cada transação envolvendo uma mesma combinação de endereços SSAP e DA, e da prioridade MAC. Dessa forma, um Componente de Transmissão existe para cada um desses casos. Em uma dada aplicação somente existirão os Componentes de Transmissão para os quais são efetivamente passadas primitivas requisitando serviços de transmissão ou troca de unidades de dados.

Cada Componente de Transmissão possui sua própria variável de estado V(SI) e seus temporizadores de confirmação e de duração da variável de transmissão. A variável V(SI) é utilizada na seleção do número de sequência de uma nova PDU a ser transmitida e no teste de sequência das PDUs de Resposta recebidas. O temporizador de confirmação determina o período de tempo durante o qual o LLC deverá ficar na espera de uma PDU de Resposta do Tipo 3 de um LLC remoto específico. O temporizador de duração da variável de transmissão define, basicamente, o tempo de existência do Componente de Transmissão.

A tabela 3.4 apresenta a tabela de transição de estados para o Componente de Transmissão.

ESTADO	EVENTO	AÇÃO	PROX_EST
IDLE	MA_UNITDATA_STATUS or RECEIVE_ACn_RSP	(Nenhuma)	IDLE
	DATA_ACK_REQUEST	SEND_ACn_CMD(SQC=V(SI), P=0) START_ACK_TIMER RETRY_COUNT:=RETRY_COUNT+1	WAIT_A
	REPLY_REQUEST	SEND_ACn_CMD(SQC=V(SI), P=1) START_ACK_TIMER RETRY_COUNT:=RETRY_COUNT+1	WAIT_R
WAIT_A	RECEIVE_ACn_RSP(SQR<>V(SI), LSDU=NULL)	DATA_ACK_STATUS_INDICATE(STATUS=STATUS_SUBFIELD) CANCEL_ACK_TIMER RETRY_COUNT:=0 V(SI):=1-V(SI)	IDLE
	RECEIVE_ACn_RSP(SQR<>V(SI), LSDU<>NULL)	DATA_ACK_STATUS_INDICATE(STATUS=PE) CANCEL_ACK_TIMER RETRY_COUNT:=0 V(SI):=1-V(SI) REPORT_STATUS(ILLEGAL_LSDU)	IDLE
	RECEIVE_ACn_RSP(SQR=V(SI)) or MA_UNITDATA_STATUS and TRANSMISSION_STATUS=GOOD	(Nenhuma)	WAIT_A
	MA_UNITDATA_STATUS and TRANSMISSION_STATUS=BAD	DATA_ACK_STATUS_INDICATE(STATUS=TRANSMISSION_STATUS) CANCEL_ACK_TIMER RETRY_COUNT=0	IDLE
	ACK_TIMER_EXPIRED and RETRY_COUNT<N4	RESEND_OLD_CMD START_ACK_TIMER RETRY_COUNT:=RETRY_COUNT+1	WAIT_A
	ACK_TIMER_EXPIRED and RETRY_COUNT>=N4	DATA_ACK_STATUS_INDICATE(STATUS=UNSUCCESSFUL) RETRY_COUNT:=0	IDLE
WAIT_R	RECEIVE_ACn_RSP(SQR<>V(SI), R=OK)	REPLY_STATUS_INDICATE(STATUS=STATUS SUBFIELD, LSDU=GIVEN LSDU) CANCEL ACK_TIMER RETRY_COUNT:=0 V(SI):=1-V(SI)	IDLE

Tabela 3.4: Tabela de Transição de Estados do CT

ESTADO	EVENTO	AÇÃO	PROX_EST
WAIT_R	RECEIVE_ACn_RSP(SQR<>V(SI), R<>OK)	REPLY_STATUS_INDICATE(STATUS=STATUS SUBFIELD, LSDU=NULL) CANCEL ACK_TIMER RETRY_COUNT:=0 V(SI):=1-V(SI)	IDLE
	RECEIVE_ACn_RSP(SQR=V(SI)) or MA_UNITDATA_STATUS and TRANSMISSION_STATUS=GOOD	(Nenhuma)	WAIT_R
	MA_UNITDATA_STATUS and TRANSMISSION_STATUS=BAD	REPLY_STATUS_INDICATE(STATUS=TRANSMISSION_STATUS, LSDU=NULL) CANCEL_ACK_TIMER RETRY_COUNT:=0	IDLE
	ACK_TIMER_EXPIRED and RETRY_COUNT<N4	RESEND_OLD_CMD START_ACK_TIMER RETRY_COUNT:=RETRY_COUNT+1	WAIT_R
	ACK_TIMER_EXPIRED and RETRY_COUNT>=N4	REPLY_STATUS_INDICATE(STATUS=UNSUCCESSFUL, LSDU=NULL) RETRY_COUNT:=0	IDLE

Tabela 3.4: Tabela de Transição de Estados do CT (cont.)

Dependendo dos serviços oferecidos pela subcamada MAC, a operação Tipo 3 do LLC pode ser simplificada tanto à nível do número de variáveis de estados mantidas nas estações, como também no número de transições encontradas nas tabelas de transição de estados. Tais simplificações serão discutidas em 4.2.3.

A existência dos Componentes de Transmissão e Recepção, com suas máquinas de protocolo distintas, permite a confecção de nós Mini-MAP com diferentes configurações. Por exemplo, sensores inteligentes que se ligam à rede local constituem-se, normalmente, em nós que somente necessitam ser lidos de tempos em tempos por um outro nó que possui o controlador do processo no qual os sensores acham-se inseridos. Neste caso, o nó correspondente ao sensor inteligente pode apresentar uma configuração simplificada, na qual o seu LLC apenas utiliza o Componente de Recepção e o MMS utiliza o serviço de preparação de dados da operação Tipo 3 do LLC, enquanto que o nó controlador do processo teria o Componente de Transmissão para as transações que envolvem a leitura dos sensores ou o envio de dados para outros nós controladores conectados à rede e, eventualmente, o Componente de Recepção para o caso de existirem transações de envio de dados de outros nós para ele.

Se o recurso de *resposta imediata* da subcamada MAC for utilizado, os nós associados aos sensores inteligentes, citados no exemplo acima, não precisam participar do anel lógico da rede local para poderem ser acessados (precisam apenas ter endereços físicos distintos dos demais nós), uma vez que um nó ao solicitar dados dos sensores, mantém o "token" em seu poder enquanto aguarda a resposta. Com o uso desses *nós não-membros do anel lógico*, melhora-se o tempo de acesso da rede local, devido à diminuição do tempo de rotação do "token" entre os nós participantes do anel.

Capítulo 4

Projeto da Interface para Rede Local (IRL)

Este projeto de uma interface de rede local foi o primeiro passo dado pelo Instituto de Automação do CTI para iniciar atividades nessa área, entendendo que as redes locais seriam de extrema importância, à médio e longo prazo, no campo de automação industrial. O principal objetivo deste projeto é o de servir como protótipo para experiências em implementações de hardware e software dedicados à redes locais para automação industrial. Ele não pretende ser um produto final, uma vez que já se conhece algumas deficiências advindas do hardware utilizado, que serão discutidas a seguir, as quais não permitem que este projeto se enquadre plenamente dentro dos requisitos desejados. Entretanto, ele tem servido para a obtenção de conhecimento nessa área e, certamente, tal experiência adquirida será de grande valia no desenvolvimento de futuros projetos, e na disseminação da tecnologia de redes locais para automação industrial.

O projeto teve início há cerca de quatro anos, porém nessa época os seus objetivos eram diferentes dos atuais [APPE87], em decorrência do próprio estágio da padronização de normas internacionais em redes locais.

No que se referia ao hardware, os circuitos integrados dedicados ao protocolo "token-passing" de acesso ao meio haviam sido lançados no mercado recentemente e, uma vez que a norma IEEE 802.4 se encontrava a nível de "draft" e pouco difundida, eles não eram confeccionados aderentes à essa norma. Existiam dois dispositivos que implementavam o protocolo "token-passing": o SMC 9026 (Standard Micro Systems) e o WD 2840 (Western Digital). A escolha recaiu sobre o WD 2840 que apesar de ter uma taxa de transmissão nominal menor que o outro (1 Mbit/seg contra 2,5 Mbit/seg do SMC 9026), era mais moderno e incorporava alguns recursos que simplificavam o projeto de hardware [CUSH84].

Quanto ao software, não se sabia exatamente como ele seria, porém desejava-se que ele tornasse a implementação aderente às normas internacionais a partir da subcamada LLC. Dessa forma, a subcamada MAC "não-padrão" foi isolada através de uma interface MAC/LLC padrão e partiu-se para a implementação de uma subcamada LLC aderente ao IEEE 802.2. Nesta subcamada, implementou-se inicialmente a operação Classe 1, por ser mais simples e

permitir que os primeiros contatos com protocolos de comunicação fossem feitos de uma forma crescente em termos de complexidade.

Foi com o surgimento da especificação MAP e, posteriormente, da especificação Mini-MAP, voltada para aplicações de redes locais que exigem tempos de resposta mais rápidos, que o projeto da IRL teve seu objetivo mais claramente definido: o de implementar um nó tipo Mini-MAP utilizando-se o hardware já desenvolvido.

A seguir serão descritos os aspectos relacionados ao desenvolvimento do hardware e do software da IRL, referentes às duas camadas inferiores do Modelo de Referência OSI (Física e Enlace). A camada de Aplicação está sendo desenvolvida por uma equipe do Instituto de Automação do CTI [APPE89] e não faz parte do escopo deste trabalho.

4.1 Projeto de Hardware da IRL

Este item apresenta a arquitetura de hardware desenvolvida para a IRL. Esta interface provê a lógica que permite ao sistema hospedeiro comunicar-se com outros sistemas conectados à rede local, utilizando um canal rápido e confiável para a transferência de dados.

4.1.1 Ambiente de Utilização da IRL

A IRL é uma placa inteligente que conecta-se ao barramento de sinais do sistema hospedeiro, sendo vista por ele como uma área de memória RAM. Este tipo de conexão caracteriza um forte acoplamento dos dois dispositivos e possibilita uma troca eficiente de informações entre eles. A figura 4.1 ilustra uma configuração básica de um sistema hospedeiro e como a IRL se conecta a ele.

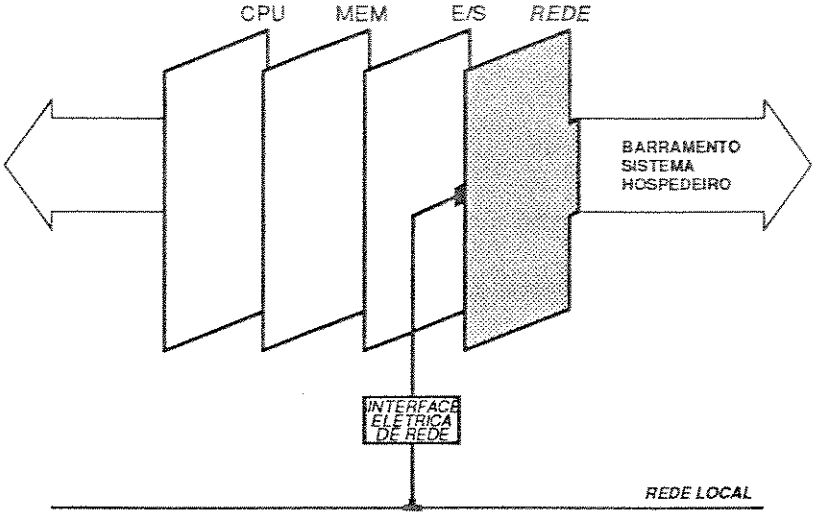

Figura 4.1: Configuração Básica de um Nó de Rede

Foram desenvolvidas placas de IRL para dois barramentos de microcomputadores distintos: o do IBM-PC/XT (e compatíveis) e o ECB (European Conector Bus). A escolha do barramento ECB resulta do fato de utilizar um conector apropriado para aplicações industriais (conector EUROPEU) e ser um padrão utilizado por, pelo menos, três fabricantes nacionais de microcomputadores industriais com uma vasta gama de produtos já desenvolvidos para essa área. A adoção do barramento IBM-PC justifica-se pelo uso cada vez maior desses microcomputadores em fábricas, quer seja em suas atividades de gestão como, também, no controle de processos. O hardware desses dois tipos de IRL é basicamente o mesmo, diferenciando apenas na lógica que opera junto a cada um dos barramentos dos sistemas hospedeiros.

A seguir são apresentadas algumas informações sobre os dois barramentos utilizados.

Barramento do Sistema IBM-PC

Para prover o computador PC com recursos adicionais tais como controlador de vídeo ou de comunicação serial, placas de interface com essas funções são conectadas ao barramento do sistema. Essas placas têm dimensões máximas de 30,48 cm de comprimento por 11,43 cm de altura e possui um conector do tipo "edge" de 62 pinos.

O barramento PC é um superconjunto do barramento do microprocessador Intel 8088. Para formar o barramento PC, aos sinais do 8088 são adicionados os sinais para suportar DMA ('Direct Memory Access"), interrupções, temporização e controle para acessos de leitura e escrita à dispositivos de E/S e memória, geração de "wait-states", "refresh" de memória e deteção de erros. Além desses, o barramento PC inclui linhas de alimentação (GND, + /-5 Vdc e + /-12 Vdc). As linhas de endereço são demultiplexadas (no 8088 as 8 linhas de dados são multiplexadas com as 8 linhas de endereços inferiores), com um espaço de endereçamento de até 1 Mbyte de memória. Ainda que o 8088 possua um espaço de até 65536 endereços de dispositivos de E/S, o PC apenas utiliza 1024 desses endereços.

O barramento PC não foi projetado para aplicações multi-mestre.

Barramento ECB

O barramento ECB possui uma mecânica similar ao barramento VME, utilizando um conector Europeu de 64 pinos e placas do tipo Eurocard simples (160 x 100 mm). As linhas de sinais são um superconjunto dos sinais do microprocessador Zilog Z80. O espaço de endereçamento de memória é de 1 Mbyte (no Z80 ele é de 64 Kbyte) e as transferências ocorrem em 8 linhas de dados segundo um protocolo síncrono. Os dispositivos de E/S são mapeados em um espaço de endereçamento com 256 posições.

Existem duas linhas de interrupção, sendo uma delas não-mascarável. A prioridade no atendimento das interrupções mascaráveis é definida por um procedimento denominado "daisy-chainning". Este barramento permite a presença de mais de um mestre.

4.1.2 Arquitetura da IRL

A figura 4.2 apresenta os módulos principais de hardware que constituem a IRL.

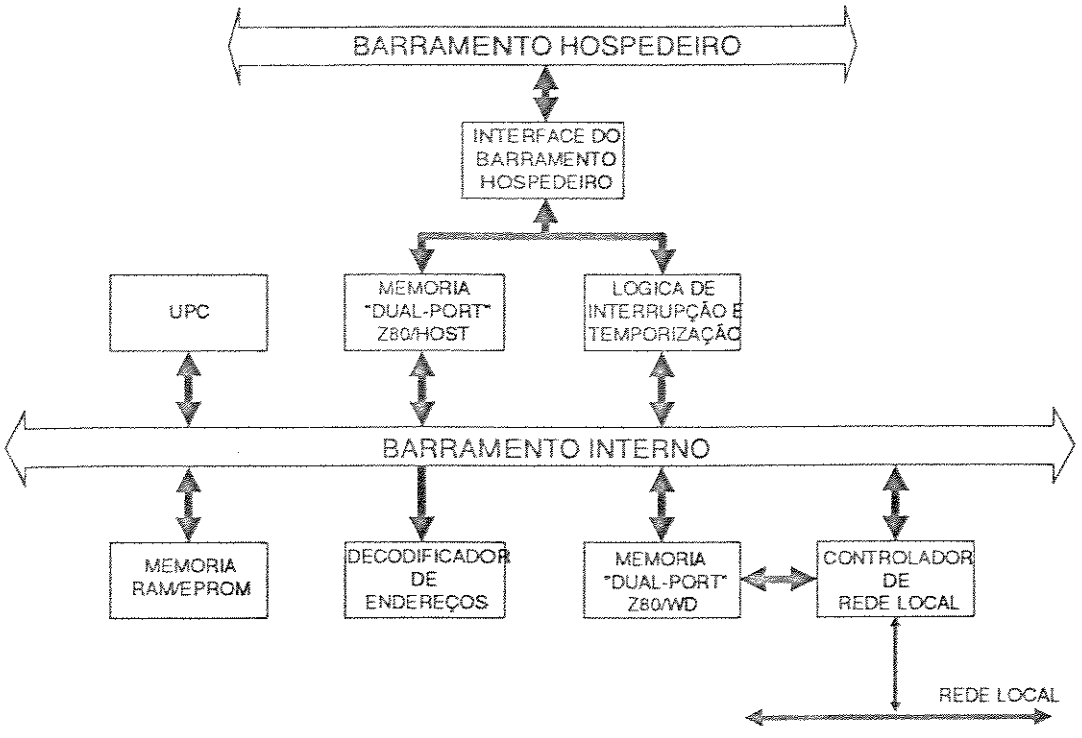

Figura 4.2: Módulos do Hardware da IRL

Nota-se dentre os módulos de hardware, a existência de duas memórias "dual-port". A utilização de memórias desse tipo permite que haja um certo grau de paralelismo nas operações realizadas pelo microprocessador interno da IRL, pelo "chip" controlador de rede e pelo hospedeiro ao realizar trocas de dados com a IRL.

A seguir são apresentadas as características principais de cada um dos módulos da IRL.

UPC (Unidade de Processamento Central)

É responsável pela inteligência da IRL, executando os programas que controlam a operação de toda a interface. Utiliza uma unidade de processamento de 8 bits, o Z80A-CPU [ZILO82], operando a 4 MHz de frequência de relógio.

Memória RAM/EPROM

Este módulo agrupa as memórias do tipo RAM e EPROM utilizadas para armazenar programas e dados da IRL. A memória EPROM contém o "firmware" da IRL, podendo

suportar até 32 Kbytes de programa. A memória RAM possui capacidade de 16 Kbytes, sendo utilizada como área de trabalho, contendo as áreas de dados e de pilha.

Memória "Dual-Port" Z80/HOST

Permite a comunicação entre a IRL e o sistema hospedeiro no qual a interface se acha conectada. Seus 8 Kbytes de capacidade são utilizados como "buffer" de transmissão e recepção de informações entre os dois dispositivos. Os acessos desta memória pelos dois dispositivos, simultâneamente, são solucionados pela lógica de árbitro de controle de acesso; foi incorporada, também, uma lógica de "lock" que permite a um dos dispositivos realizar múltiplos acessos à esta memória, como parte de uma operação "atômica".

Memória "Dual-Port" Z80/WD

Possue também 8 Kbytes de capacidade, utilizados como "buffer" de transmissão e recepção dos "frames" enviados/recebidos pelo controlador da rede local (WD 2840). Além disso, uma área desta memória é utilizada para armazenar parâmetros de iniciação do controlador da rede, assim como dados estatísticos de operação da rede local [WEST84].

Interface do Barramento Hospedeiro

É composta, basicamente, por circuitos integrados do tipo "buffer" e "transceiver" ligados às linhas de dados, endereço e controle do barramento do sistema hospedeiro. É constituída, também, por uma lógica de decodificação de endereços da memória "dual-port" Z80/HOST e de dispositivos de E/S (utilizados pelo sistema hospedeiro para gerar interrupções ao Z80 da IRL e na operação da lógica de "lock" da memória "dual-port" Z80/HOST).

Lógica de Interrupção e Temporização

Este módulo é implementado utilizando-se um circuito integrado Z80A-CTC [ZILO82] e tem como finalidade gerar interrupções ao Z80 da IRL na ocorrência dos seguintes eventos: pedidos de interrupção do controlador da rede local, do sistema hospedeiro e do gerador de base de tempo para o software da IRL.

Decodificador de Endereços

Possue a lógica responsável pelo mapeamento da área de memória e dos dispositivos de E/S presentes na IRL. A lógica é formada, basicamente, por PROMs ("Programmable Read Only Memory") que realizam a decodificação de endereços.

Controlador da Rede Local

Este módulo é composto, essencialmente, pelos circuitos integrados WD 2840 e HD 6409, além de uma interface elétrica de comunicação padrão RS-485/422.

O WD 2840 [WEST84] é o controlador da rede local propriamente dito, responsável pela execução do protocolo "token-passing" de acesso ao meio físico da rede. Suas funções internas liberam o processador da IRL da complexa tarefa de implementar esse protocolo e

permite que ele seja utilizado na execução dos protocolos de mais alto nível. Algumas das suas principais características são:

- taxa de transmissão de 1 Mbit/seg;
- permite até 254 estações na rede;
- suporte para transmissões do tipo "broadcast";
- opção de confirmação para cada "frame" transmitido;
- geração de dados estatísticos de operação da rede.

O HD 6409 [HARR84] é um codificador/decodificador de sinais do tipo "Manchester". Ele codifica os pulsos NRZ ("Non Return to Zero"), transmitidos pelo WD 2840, para pulsos "Manchester" antes de enviá-los à interface RS-485/422 e executa a operação inversa nos sinais recebidos dessa interface.

A interface elétrica padrão RS-485/422 possui dois empregos distintos: quando a IRL se liga diretamente ao meio físico da rede local e quando a IRL se liga ao meio físico através de um módulo "modem" (que permite a implementação de outros tipos de meios físicos:"carrierband", "broadband", etc). No primeiro caso, a interface é dotada de circuitos integrados que implementam o padrão RS-485, apropriados para a arquitetura "token-bus" da rede local, enquanto que no segundo caso ela é dotada de circuitos integrados do padrão RS-422, apropriados para interligar dispositivos que se comunicam com taxas de transmissão altas e se acham afastados fisicamente.

4.2 Projeto de Software da IRL

Tendo-se em vista o desenvolvimento de um nó de rede local coerente com a arquitetura Mini-MAP e levando-se em consideração as limitações do hardware utilizado, optou-se pela separação física das camadas que compõem essa arquitetura, como mostra a figura 4.3.

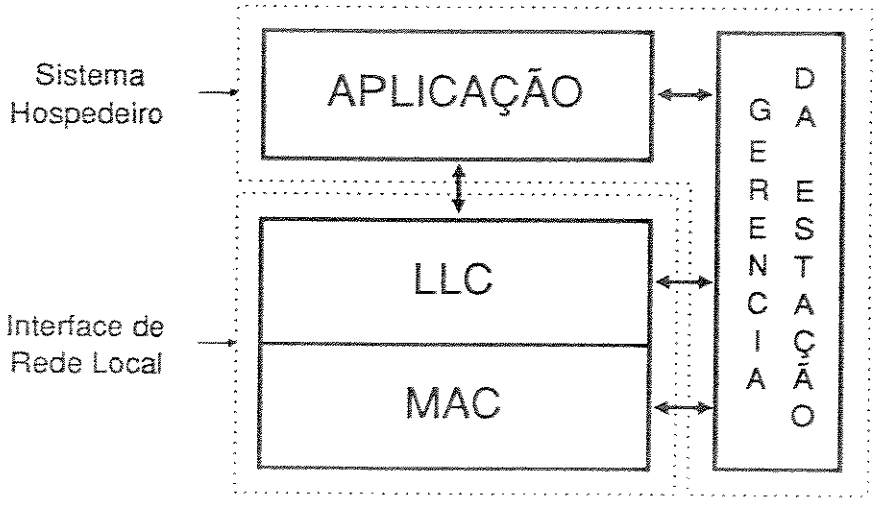

Figura 4.3: Separação das Camadas no Nó de Rede

Assim, a camada Física e a camada de Enlace ficariam localizadas na IRL, enquanto que a camada de Aplicação e a entidade de Gerência da Estação se localizariam no sistema hospedeiro. Essa separação física das camadas implica no desenvolvimento de uma interface própria, tornando possível a interação entre elas; neste projeto, essa interação ocorre através de uma memória "dual-port" da IRL, acessível pelo sistema hospedeiro, a qual permite a troca de primitivas codificadas entre as duas camadas separadas. O envio e a recepção das primitivas por esta interface são sincronizadas pela geração de interrupções, isto é, o sistema hospedeiro gera uma interrupção na IRL após ter colocado uma primitiva na memória "dual-port", e vice-versa.

Das duas camadas presentes na IRL, a camada Física e grande parte da subcamada MAC, são implementadas por hardware. A camada Física é representada, basicamente, pelos cabos, conectores, circuitos da interface padrão RS-485, codificador Manchester e sinais da lógica de transferência de dados localizada no dispositivo controlador de rede local. Este mesmo componente é responsável pela realização das funções necessárias ao controle de acesso ao meio físico da rede local, englobando a maior parte das operações definidas na subcamada MAC do tipo "token-bus".

O projeto de software da IRL resultou, portanto, na elaboração da subcamada LLC em sua totalidade e de uma pequena porção da subcamada MAC. Os programas feitos para a subcamada MAC criaram a interface LLC/MAC definida na norma IEEE 802.4, de forma que a subcamada LLC pudesse requisitar serviços através de procedimentos bem definidos, que seriam, posteriormente, executados pelo MAC satisfazendo os modos de operação do dispositivo controlador de rede local.

Além das interfaces Aplicação/LLC e LLC/MAC, foram implementadas as interfaces Gerência/MAC e Gerência/LLC. Tais interfaces são necessárias para que a entidade de Gerência da Estação possa, por exemplo, ler e alterar parâmetros de operação do MAC ou criar SAPs no LLC.

Para que o usuário da IRL localizado no sistema hospedeiro não se preocupe com as peculiaridades da interface física entre os dois dispositivos, é fornecido um conjunto de funções que efetivamente implementam as interfaces Aplicação/LLC, Gerência/MAC e Gerência/LLC. Essas funções são dependentes do sistema hospedeiro e do sistema operacional por ele utilizado.

4.2.1 Software Básico

O software desenvolvido para a IRL teve como objetivo a criação de uma filosofia de implementação de protocolos para redes locais em aplicações industriais. Essa implementação baseou-se na utilização de um *ambiente multi-tarefas*, o qual permitiu que as necessidades desse tipo de aplicação pudessem ser satisfeitas. Essas necessidades traduzem-se, basicamente, no conceito de concorrência entre programas. Isso porque, da mesma forma que programas de aplicação em tempo-real são executados nos sistemas hospedeiros competindo pelos recursos computacionais, os programas que tratam as

transferências de dados pela rede local, geradas por esses aplicativos, devem também competir pelos recursos computacionais da IRL.

A introdução de um executivo de tempo-real possibilita a implementação de protocolos de comunicação que suportam mensagens com prioridades distintas, necessárias nas aplicações industriais. Pode-se tomar como exemplo o caso em que uma transação de envio de dados de um programa aplicativo normal está sendo realizada pelos programas de comunicação; em um dado instante, uma mensagem de alarme de altíssima prioridade deve ser enviada à rede local, a fim de se evitar um acidente de grandes proporções. Nesse instante, o executivo de tempo-real é responsável em fornecer as condições que possibilitem a essa mensagem de alta prioridade ser enviada antes de qualquer outra; neste caso, ele deve suspender o processamento da mensagem corrente e ativar os programas que tratarão do envio da nova mensagem.

Processos industriais possuem restrições de tempo bastante rígidas se comparados a outros tipos de aplicações e, portanto, precisam ser processados de uma maneira capaz de diferenciar suas atividades mais prioritárias. Assim, a implementação de um software de comunicação baseado na concorrência entre tarefas se faz necessária no sentido de contemplar esses requisitos.

O ambiente multi-tarefas utilizado neste projeto foi implementado a partir de uma adaptação do sistema operacional XINU [COME84], para o microprocessador Z80. Este sistema operacional foi implementado segundo uma estrutura de camadas como mostra a figura 4.4. Esta organização não descreve o fluxo de controle no sistema operacional, mas apenas a forma na qual a sua implementação se acha estruturada.

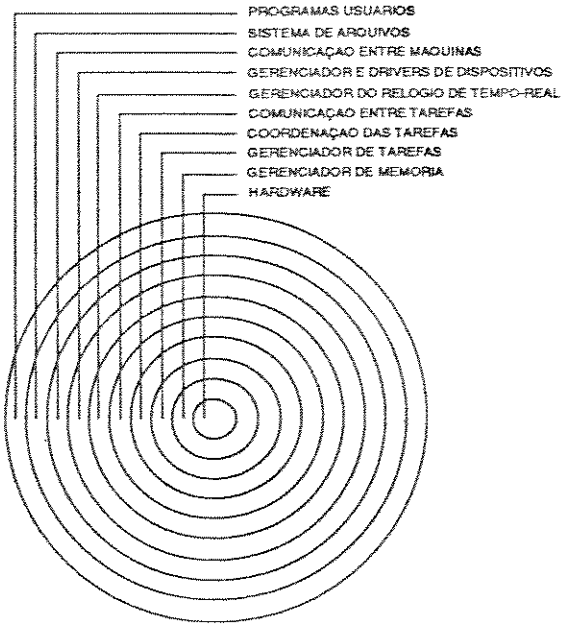

Figura 4.4: Estrutura do Sistema Operacional XINU

Neste projeto utilizou-se somente uma configuração mínima, a qual abrange da camada mais interna até a camada *gerenciador do relógio de tempo-real*, e a parte da camada *comunicação entre máquinas* que trata da implementação de *filas* ("ports"). Esta configuração será denominada, daqui por diante, por NMT (Núcleo Multi-Tarefas).

O NMT é implementado usando-se a linguagem de programação C, a menos da função que realiza a troca de contexto de tarefas, a qual é feita em linguagem "assembly" do Z80.

As *chamadas ao sistema* ("system calls") fornecidas pelas camadas selecionadas executam as seguintes funções principais:

- gerenciamento da memória compartilhada entre tarefas;
- criação e ativação de tarefas;
- gerenciamento de comunicação entre tarefas;
- gerenciamento de semáforos;
- gerenciamento de temporizações;
- gerenciamento da prioridade das tarefas.

As chamadas ao sistema são realizadas através de funções escritas na linguagem C. A seguir são descritas as funções utilizadas neste projeto.

Gerenciamento de Memória Compartilhada entre Tarefas

- *getmem(nbytes)*: aloca um bloco de *nbytes* de memória da área de "heap";
- *freemem(ptr, size)*: libera o bloco de memória de tamanho *size*, apontada por *ptr*, para a área de "heap".

Criação e Ativação de Tarefas

- *create(caddr, ssize, prio, name, nargs[,argument])*: cria uma nova tarefa que iniciará execução no endereço *caddr*, com um "stack" de *ssize* palavras, com prioridade inicial *prio*, com nome de identificação *name*, recebendo *nargs* argumentos passados na lista *argument*. O processo criado é colocado no estado *suspenso*.
- *resume(pid)*: tira a tarefa *pid* do estado *suspenso* e coloca-a no estado *pronto* para executar;
- *kill(pid)*: remove a tarefa *pid* do sistema.

Gerenciamento de Comunicação entre Tarefas

- *pcreate(count)*: cria uma fila com *count* posições para armazenar ponteiros de mensagens;
- *psend(portid, message)*: envia uma mensagem (ou seja, o seu ponteiro *message*) para a fila *portid*;
- *preceive(portid)*: retira uma mensagem da fila *portid*;
- *pcount(portid)*: retorna o número de mensagens existentes na fila *portid*.

Gerenciamento de Semáforos

- *screate(count)*: cria um semáforo contador e inicializa-o com o valor *count*;
- *signal(sem)*: libera uma tarefa que se encontra bloqueada no semáforo *sem*, ou, caso não exista tarefa bloqueada, incrementa o contador deste semáforo;
- *wait(sem)*: decrementa o contador do semáforo *sem*, bloqueando a tarefa chamadora se o valor do contador ficar negativo.

Gerenciamento de Temporizações

- *sleep(secs)*: coloca tarefa chamadora para dormir por *secs* segundos (usado para "delays" longos);
- *sleep10(tenths)*: coloca tarefa chamadora para dormir por *tenths* décimos de segundo (usado para "delays" curtos).

Gerenciamento da Prioridade das Tarefas

- *chprio(pid, newprio)*: muda a prioridade da tarefa *pid* para o valor passado em *newprio*;
- *getprio(pid)*: obtém a prioridade da tarefa *pid*.

O NMT opera no modo *preemptivo*, sendo que a característica adotada na implementação original do XINU de tratar tarefas de mesma prioridade em um esquema do tipo "round-robin" foi retirada, ou seja, o processamento de uma tarefa não é mais interrompido por uma outra tarefa de mesma prioridade.

4.2.2 Descrição das Interfaces Lógicas entre Camadas

As interfaces entre as camadas de protocolos de comunicação constituem-se no meio pelo qual uma camada requisita ou fornece serviços a uma outra camada vizinha. Esses serviços são requisitados ou fornecidos através da utilização de primitivas de serviços, as quais são distintas para cada uma das interfaces e dependem do tipo de operação realizada em cada camada. A figura 4.5 mostra as interfaces implementadas neste projeto.

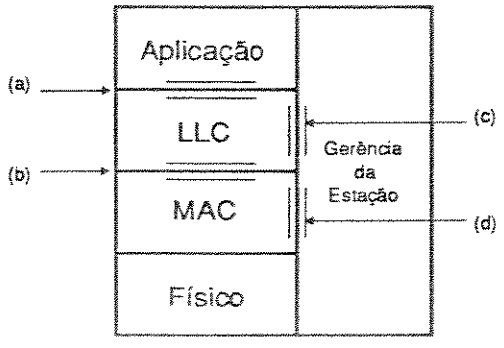

Figura 4.5: Interfaces Utilizadas no Projeto

Interface Aplicação/LLC (a)

Por operar em Classe 3, a subcamada LLC oferece à camada de Aplicação os serviços Tipo 1 e Tipo 3, descritos a seguir:

1. Serviços do Tipo 1

Esses serviços permitem a troca de unidades de dados entre entidades das camadas de Aplicação pares, sem o estabelecimento de uma conexão de enlace e sem confirmação. Os serviços Tipo 1 são usados nas transferências de dados do tipo "broadcast", "multicast" e nas da Gerência da Estação. As primitivas associadas a este tipo de operação são [IEEE84]:

- L_DATA.request
- L_DATA.indication

2. Serviços do Tipo 3

Permitem a troca de unidades de dados entre entidades das camadas de Aplicação pares, sem o estabelecimento de uma conexão de Enlace, porém, com confirmação. Estes serviços são utilizados em todas as transferências de dados, exceto naquelas onde os serviços Tipo 1 são exigidos. As primitivas associadas com este tipo de operação são [ISO86]:

- Primitivas para transmissão de dados

- DL_DATA_ACK request
- DL_DATA_ACK indication
- DL_DATA_ACK_STATUS indication

- Primitivas para troca de dados

- DL_REPLY request
- DL_REPLY indication
- DL_REPLY_STATUS indication

- Primitivas para preparação de dados

- DL_REPLY_UPDATE request
- DL_REPLY_UPDATE_STATUS indication

Interface LLC/MAC (b)

Define as primitivas que permitem a transferência de dados entre entidades LLC pares, sem o estabelecimento de uma conexão ponto-a-ponto entre subcamadas MAC pares [IEEE85]. As primitivas são:

- MA_DATA.request
- MA_DATA.indication
- MA_DATA.confirmation

Interface Gerência/LLC (c)

Na falta de um documento ISO ou IEEE que descrevesse esta interface, na época do desenvolvimento deste projeto, foi adotado o documento da ISA ("Instruments Society of America") citado em [ISA88].

As primitivas que definem esta interface estão relacionadas, basicamente, com a ativação, manutenção e desativação dos SAPs da subcamada LLC. As primitivas são:

- L_RESET.request
- L_SAP_ACTIVATE.request
- L_SAP_ACTIVATE.confirmation
- L_SAP_DEACTIVATE.request
- L_SAP_DEACTIVATE.confirmation
- L_STATUS.request
- L_STATUS.indication
- L_STATUS.confirmation
- L_RSAP_ACTIVATE.request
- L_RSAP_ACTIVATE.confirmation

Interface Gerência/MAC (d)

As primitivas desta interface estão relacionadas, basicamente, com a iniciação, leitura e notificação de alteração de parâmetros relevantes para a operação da subcamada MAC, tais como: endereço da estação, contadores e temporizadores [IEEE85]. As primitivas são:

- MA_INITIALIZE_PROTOCOL.request
- MA_INITIALIZE_PROTOCOL.confirmation
- MA_SET_VALUE.request
- MA_SET_VALUE.confirmation
- MA_READ_VALUE.request
- MA_READ_VALUE.confirmation
- MA_EVENT.indication
- MA_FAULT_REPORT.indication
- MA_GROUP_ADDRESS.request
- MA_GROUP_ADDRESS.confirmation
- MA_CDATA.request
- MA_CDATA.indication
- MA_CDATA.confirmation

4.2.3 Particularidades do Projeto de Software

Por utilizar um hardware disponível cujas limitações eram conhecidas no que se refere à utilização desejada, o desenvolvimento do software da IRL teve quatro particularidades principais, as quais são discutidas a seguir.

Utilização de Memória Interna da IRL

Os programas executando na IRL têm somente 16 Kbytes de memória RAM para serem utilizados como memória de trabalho. Como esse espaço de memória deve ser usado basicamente para acomodar as estruturas de dados e as áreas de "stack" das tarefas criadas, optou-se em não utilizá-lo no armazenamento temporário dos dados recebidos da rede local ou do sistema hospedeiro; estes serão sempre mantidos nas áreas das memórias "dual-port" onde foram colocados até que sejam completamente tratados pelo software da IRL.

Essa opção foi feita não somente devido à escassez de memória de trabalho, como também pelo fato de se estar utilizando, no NMT, um gerenciamento de memória simples, sobre um único "buffer pool". No caso de um fluxo intenso de dados, esse gerenciamento não seria suficiente para evitar um "estouro" da memória de trabalho. O XINU possui um esquema mais complexo de gerenciamento de memória que poderia ser empregado no projeto, porém não o foi por causar um aumento significativo no código dos programas da IRL.

Por outro lado, a área destinada às memórias "dual-port", não gerenciada pelo NMT, possui uma boa capacidade de armazenamento de mensagens, permitindo uma permanência mais prolongada delas no "buffer" de transmissão ou recepção em que se encontram. Além disso, essas memórias constituem-se em "buffers" finitos usados pelo LLC e MAC. Particularmente, no caso da interface Aplicação/LLC, a existência desse tipo de "buffer" na memória "dual-port" Z80/HOST é bastante útil, por contribuir no controle de fluxo das primitivas trocadas nesta interface, uma vez que essas trocas são feitas de forma assíncrona [SVOB89].

Utilização de Prioridade nas Transferências de Dados

Apesar do controlador de rede WD 2840 não suportar um esquema de prioridades a nível da subcamada MAC, como recomenda a especificação Mini-MAP, a implementação da subcamada LLC adotou essa opção. Dessa forma, gerou-se uma implementação do LLC contemplando a concorrência entre as tarefas que tratam as transferências de mensagens com prioridades distintas.

Adição do Campo FC no "Frame" do MAC

Para que o emprego de prioridades fosse possível a nível da rede local como um todo, foi preciso adicionar um novo campo no "frame" gerado pelo controlador da rede, denominado FC ("Frame Control"), como mostra a figura 4.6. A criação deste campo foi baseada no formato do "frame" utilizado pela norma IEEE 802.4 e ele transporta, além da prioridade, a informação do tipo de "frame" transmitido, ou seja, indica se é um "frame" de controle do MAC ou um "frame" de dados.

O "frame" de controle do MAC é utilizado pelo procedimento de adição de novas estações ao anel lógico da rede e utiliza o campo FC para que seu tratamento pelo software da IRL seja distinto ao do "frame" de dados, que é utilizado na transferência normal de dados.

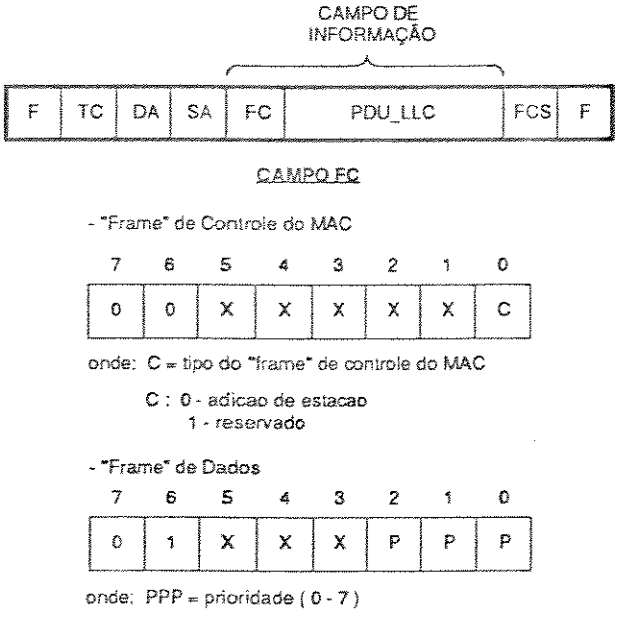

Figura 4.6: Descrição do Campo FC

Simplificação das Máquinas de Estados da Subcamada LLC

O Mini-MAP adota para a subcamada LLC especificações equivalentes à subcamada PLC do PROWAY [ISA88]. A especificação PROWAY impõe algumas condições específicas em relação à operação Tipo 3 genérica, merecendo destaque no documento [ISO86], na forma do Apêndice A. Essas condições, apresentadas no item A1 desse apêndice, são as seguintes:

1. A camada de acesso ao meio utilizada é o ISO DIS 8802/4 (IEEE 802.4), suportando o recurso de *resposta imediata*.
2. O serviço *DL_REPLY* é usado somente para requisitar dados, de forma que uma PDU de Comando ACn sempre terá seu campo de informação nulo quando o P bit for 1.
3. A classe de serviço MAC *"Request-with-Response"* é usada em todas as PDUs de Comando do Tipo 3.
4. O LLC faz apenas uma tentativa de enviar cada PDU de Comando do Tipo 3.
5. O Temporizador *Confirmação do Tipo 3* é eliminado.
6. O Temporizador *"receive variable lifetime"* é eliminado.
7. O Temporizador *"transmit variable lifetime"* é eliminado.

Dessas condições específicas, as de número 1, 3 e 5 não puderam ser usadas no projeto, devido ao MAC empregado. Além de não ser compatível com o padrão mencionado, o recurso de resposta imediata do controlador WD2840 também difere do especificado nessa norma por ser, na verdade, um recurso de confirmação imediata realizado a nível do controlador de rede, sem intervenção do processador da IRL. Entretanto, o recurso de confirmação imediata do WD2840 é utilizado em todas as transferências de dados do Tipo 3.

O Temporizador de Confirmação não pôde ser eliminado, novamente pelo fato de não se ter um MAC com o recurso de resposta imediata. Esse recurso permitiria ao LLC eliminar a temporização, a qual seria substituída pelo temporizador de confirmação do MAC.

Como consequência da simplificação parcial da máquina de estados do LLC, discutida acima, foram geradas duas novas tabelas de transição de estados, que resultam em um caso intermediário entre as tabelas do LLC Tipo 3 genérico e as do PROWAY. A tabela 4.1 mostra a tabela de transição de estado do Componente de Recepção e a tabela 4.2, a do Componente de Transmissão, utilizadas neste projeto.

	ESTADO	EVENTO	AÇÃO	PROX_EST
01	READY	REPLY_UPDATE_REQUEST	SAVE:=GIVEN_LSDU REPLY_UPDATE_STATUS_INDICATE	READY
02	READY	RECEIVE_ACn_CMD(SQC=V(RI), P=0, INFO<>NULL) and RECEIVE_STATUS()=OK	SEND_ACn_RSP(SQR=1-SQC, F=0, C=OK, R=NR, LSDU=NULL) DATA_ACK_INDICATE V(RI):=1-SQC V(RB):=OK	READY
03	READY	RECEIVE_ACn_CMD(SQC=V(RI), P=0, INFO=NULL) and RECEIVE_STATUS()=OK	SEND_ACn_RSP(SQR=1-SQC, F=0, C=OK, R=NR, LSDU=NULL) V(RI):=1-SQC V(RB):=OK	READY
04	READY	RECEIVE_ACn_CMD(SQC=V(RI), P=0) and RECEIVE_STATUS()<>OK	SEND_ACn_RSP(SQR=1-SQC, F=0, C=RECEIVE_STATUS(), R=NR, LSDU=NULL) V(RI):=1-SQC V(RB):=RECEIVE_STATUS()	READY
05	READY	RECEIVE_ACn_CMD(SQC=V(RI), P=1) and ACCESS()=OK	SEND_ACn_RSP(SQR=1-SQC, F=1, C=OK, R=OK, LSDU=SAVE) REPLY_INDICATE(LSDU=NULL) V(RI):=1-SQC V(RB):=OK	READY
06	READY	RECEIVE_ACn_CMD(SQC=V(RI), P=1) and ACCESS()<>OK	SEND_ACn_RSP(SQR=1-SQC, F=1, C=OK, R=ACCESS(), LSDU=NULL) V(RI):=1-SQC V(RB):=OK	READY
07	READY	RECEIVE_ACn_CMD(SQC<>V(RI), P=0)	SEND_ACn_RSP(SQR=1-SQC, F=0, C=V(RB), R=NR. LSDU=NULL)	READY
08	READY	RECEIVE_ACn_CMD(SQC<>V(RI), P=1) and ACCESS()=OK	SEND_ACn_RSP(SQR=1-SQC, F=1, C=V(RB), R=OK, LSDU=SAVE) REPLY_INDICATE(LSDU=NULL)	READY
09	READY	RECEIVE_ACn_CMD(SQC<>V(RI), P=1) and ACCES()<>OK	SEND_ACn_CMD(SQR=1-SQC, F=1, C=V(RB), R=ACCESS(), LSDU=NULL)	READY

Tabela 4.1: Tab. de Trans. de Est. do CR Adotada no Projeto

	ESTADO	EVENTO	AÇÃO	PROX_EST
01	IDLE	MA_UNITDATA_STATUS	(NENHUMA)	IDLE
02	IDLE	RECEIVE_ACn_RSP	(NENHUMA)	IDLE
03	IDLE	DATA_ACK_REQUEST	SEND_ACn_CMD(SOC=V(SI),P=0) START_ACK_TIMER	WAIT_A
04	IDLE	REPLY_REQUEST	SEND_ACn_CMD(SOC=V(SI),P=1) START_ACK_TIMER	WAIT_R
05	WAIT_A	RECEIVE_ACn_RSP(SQR<>V(SI), LSDU=NULL)	DATA_ACK_STATUS_INDICATE(STATUS=STATUS_SUBFIELD) CANCEL_ACK_TIMER V(SI):=1-V(SI)	IDLE
06	WAIT_A	RECEIVE_ACn_RSP(SQR<>V(SI), LSDU<>NULL)	DATA_ACK_STATUS_INDICATE(STATUS=PE) CANCEL_ACK_TIMER V(SI):=1-V(SI) REPORT_STATUS(ILEGAL_LSDU)	IDLE
07	WAIT_A	RECEIVE_ACn_RSP(SQR=V(SI))	(NENHUMA)	WAIT_A
08	WAIT_A	MA_UNITDATA_STATUS and TRANSMISSION_STATUS=GOOD	(NENHUMA)	WAIT_A
09	WAIT_A	MA_UNITDATA_STATUS and TRANSMISSION_STATUS=BAD	DATA_ACK_STATUS_INDICATE(STATUS=TRANSMISSION_STATUS) CANCEL_ACK_TIMER	IDLE
10	WAIT_A	ACK_TIMER_EXPIRED	DATA_ACK_STATUS_INDICATE(STATUS=UNSUCCESFUL)	IDLE
11	WAIT_R	RECEIVE_ACn_RSP(SQR<>V(SI), R=OK)	REPLY_STATUS_INDICATE(STATUS=STATUS_SUBFIELD, LSDU=GIVEN_LSDU) CANCEL_ACK_TIMER V(SI):=1-V(SI)	IDLE
12	WAIT_R	RECEIVE_ACn_RSP(SQR<>V(SI), R<>OK)	REPLY_STATUS_INDICATE(STATUS=STATUS_SUBFIELD, LSDU=NULL) CANCEL_ACK_TIMER V(SI):=1-V(SI)	IDLE

Tabela 4.2: Tab. de Trans. de Est. do CT Adotada no Projeto

	ESTADO	EVENTO	AÇÃO	PROX_EST
13	WAIT_R	RECEIVE_ACn_RSP(SQR=V(SI))	(NENHUMA)	WAIT_R
14	WAIT_R	MA_UNITDATA_STATUS and TRANSMISSION_STATUS=GOOD	(NENHUMA)	WAIT_R
15	WAIT_R	MA_UNITDATA_STATUS and TRANSMISSION_STATUS=BAD	REPLY_STATUS_INDICATE(STATUS=TRANSMISSION_STATUS, LSDU=NULL) CANCEL_ACK_TIMER	IDLE

Tabela 4.2: Tab. Trans. Est. do CT Adotada no Projeto (cont)

4.2.4 Software Aplicativo

Por software aplicativo entende-se os módulos de programas que operam na IRL e que são responsáveis pela execução dos protocolos nela implementados. Os módulos que se interrelacionam foram agrupados em tarefas para operarem no ambiente multi-tarefas citado em 4.2.1.

Essas tarefas são ativadas a partir de interrupções geradas pelos periféricos da IRL: controlador da rede, gerador de base de tempo do NMT e gerador de interrupção do hospedeiro. A única exceção fica para a *tarefa nula* criada na ativação do NMT, a qual é responsável pela iniciação das variáveis globais do sistema e pela criação de semáforos, filas de comunicação e tarefas permanentes do sistema. Semáforos, filas e tarefas podem ser criados dinamicamente em função da execução dos programas da IRL, como será visto mais adiante.

4.2.4.1 Descrição dos Armazenadores de Dados

Neste item são descritas as estruturas de dados utilizadas na implementação do software da IRL. Para se ter uma visão geral de como essas estruturas de dados se interrelacionam com as tarefas, a figura 4.7 mostra o *diagrama de dados* [MART86] do sistema, onde os retângulos representam as *estruturas de dados* (armazenadores), os hexágonos representam as *tarefas* e as setas, os *fluxos de dados.*

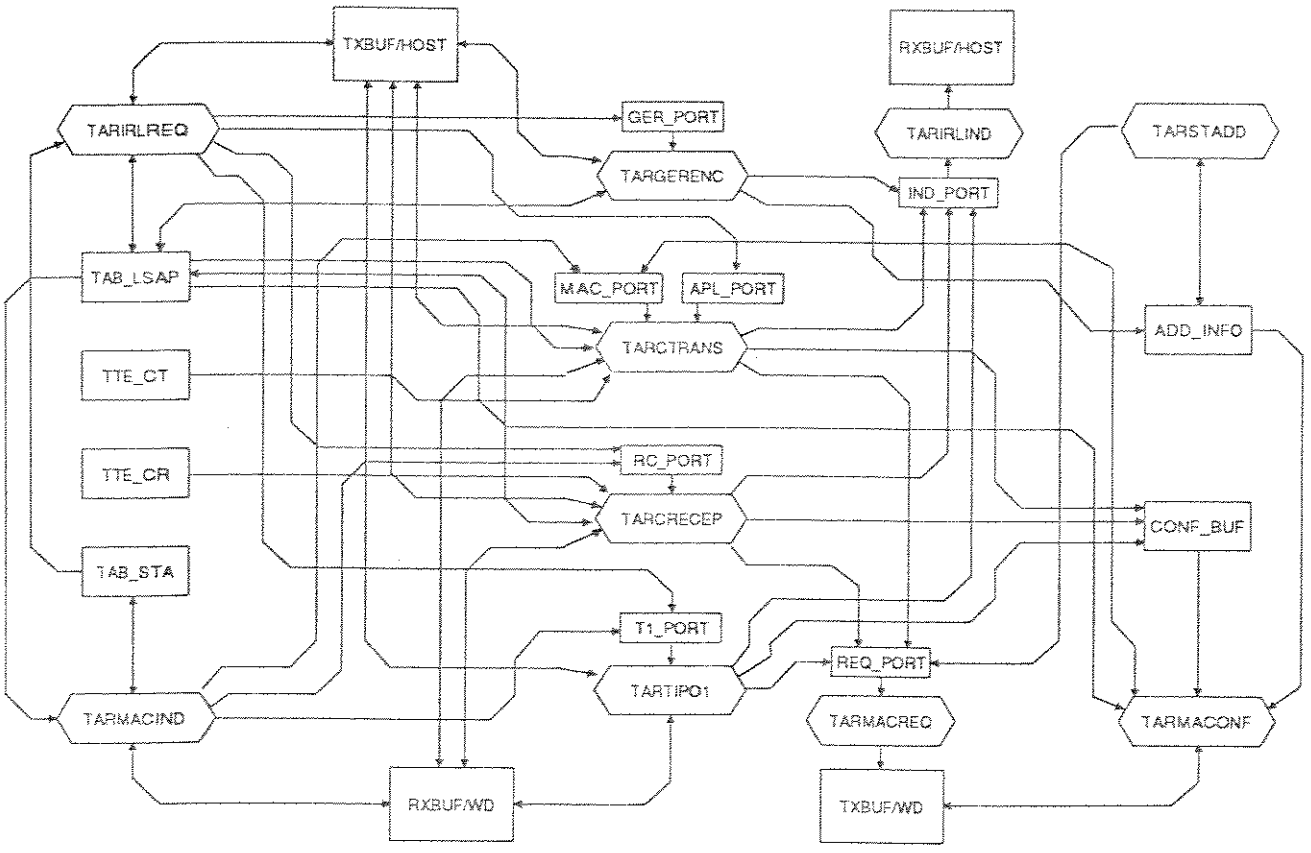

Figura 4.7: Diagrama de Dados do Software da IRL

A seguir, cada um dos armazenadores são apresentados e seus detalhamentos feitos usando-se a representação em *diagramas de uso de dados* (DUD) [CTIA84].

TXBUF/HOST

Compreende a área da memória "dual-port" Z80/HOST destinada ao armazenamento temporário das primitivas geradas pelo hospedeiro. Ela é dividida em 8 sub-áreas, sendo que cada uma delas está relacionada a um nível da prioridade MAC associada às primitivas de transferência de dados. Em uma mesma sub-área, as primitivas são colocadas segundo uma fila circular. Este esquema permite um melhor aproveitamento de memória, uma vez que as primitivas são de tamanhos variáveis, e é de operação simples. No início de cada sub-área existem dois apontadores, denominados COLOCA e RETIRA, os quais são compartilhados tanto pelo hospedeiro como pela IRL e são utilizados no gerenciamento da fila circular dessa sub-área.

Existe, ainda, uma posição de memória em TXBUF/HOST utilizada como "flag" para sincronizar a geração de interrupções do hospedeiro à IRL, após a colocação de uma primitiva neste armazenador. Isso se faz necessário para evitar que interrupções muito próximas possam causar a perda de primitivas por parte da IRL. Após a colocação de uma primitiva em TXBUF/HOST, o hospedeiro deverá verificar o estado desse "flag" para gerar a interrupção à IRL; caso ele esteja ativado, o hospedeiro deve aguardar até que a IRL encaminhe o tratamento da interrupção anterior e desative o "flag". Ao verificar que ele foi desativado, o hospedeiro deverá gerar a interrupção na IRL e ativar o "flag" novamente.

A figura 4.8 mostra a estrutura do TXBUF/HOST.

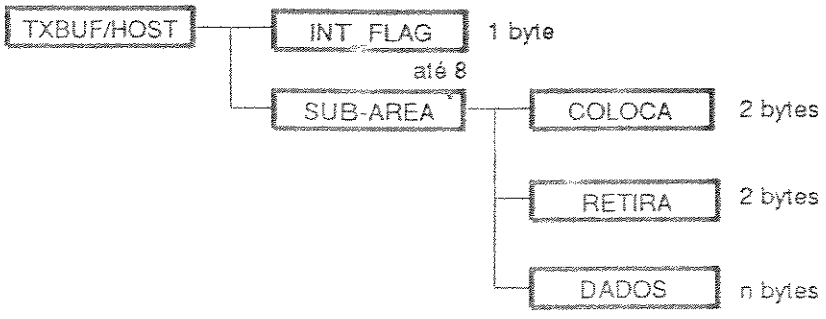

Figura 4.8: Estrutura do TXBUF/HOST

A área de dados do TXBUF/HOST abriga as primitivas enviadas pelo hospedeiro. Essas primitivas são colocadas segundo um formato que possui estreita semelhança com as suas sintaxes definidas nos padrões utilizados. Como exemplo, a figura 4.9 apresenta a conversão de uma primitiva para o formato usado em TXBUF/HOST.

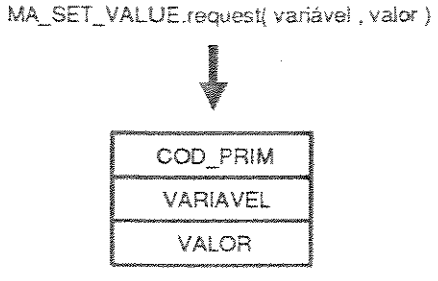

Figura 4.9: Formato de uma Primitiva Comum

Neste caso, o mapeamento da primitiva é direto. O nome da primitiva é convertido para um código inteligível pelo sistema e colocado em COD_PRIM, enquanto que os parâmetros

VARIÁVEL e VALOR são os mesmos da primitiva. As primitivas utilizadas na troca de unidades de dados de serviço (SDU), entretanto, não são mapeadas diretamente; elas necessitam de um campo a mais, o SDU_LGTH, o qual fornece o tamanho da SDU, para tornar o processamento mais simples. A figura 4.10 ilustra este caso.

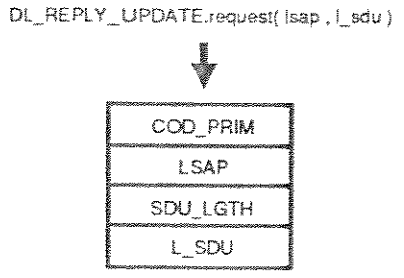

Figura 4.10: Formato de uma Primitiva de Dados

RXBUF/HOST

Compreende a área da memória "dual-port" Z80/HOST, destinada ao armazenamento temporário das primitivas enviadas ao hospedeiro, oriundas das subcamadas MAC e LLC. Como pode ser visto na figura 4.11, a estrutura do RXBUF/HOST é idêntica à do TXBUF/HOST e, os campos nele presentes possuem as mesmas funções descritas no item anterior.

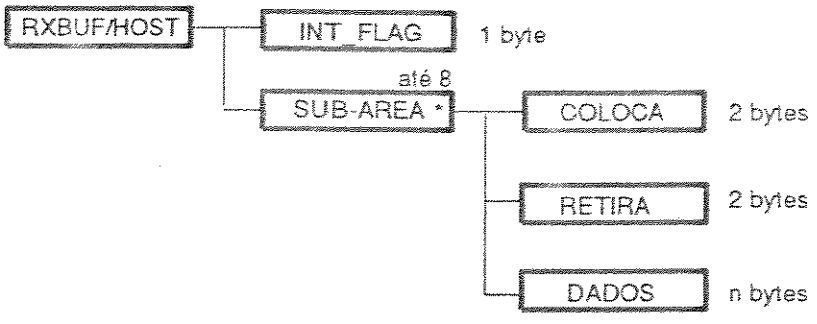

Figura 4.11: Formato do RXBUF/HOST

O formato das primitivas armazenadas no campo de dados de cada sub-área seguem as mesmas regras descritas para TXBUF/HOST.

TXBUF/WD

Compreende a área da memória "dual-port" Z80/WD, utilizada no armazenamento temporário dos "frames" a serem transmitidos para a rede local. Sua estrutura é a de uma fila circular de segmentos de memória ligados, como mostra a figura 4.12, e é definida para satisfazer o modo de operação do controlador de rede local WD 2840.

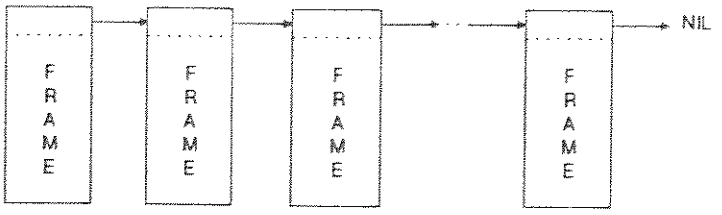

Figura 4.12: Formato do TXBUF/WD

O tamanho dos segmentos é fixo e definido na iniciação do controlador de rede local, podendo ter um mínimo de 64 bytes e um máximo de 1024 bytes. O TXBUF/WD contém um número inteiro desses segmentos, sendo que esse número depende da área de memória alocada para ele e do tamanho selecionado para os segmentos.

Um "frame" poderá ocupar um único segmento ou mais de um, até o máximo de 16 segmentos. A título de exemplo, a figura 4.13 mostra como um "frame" de 2 segmentos seria composto.

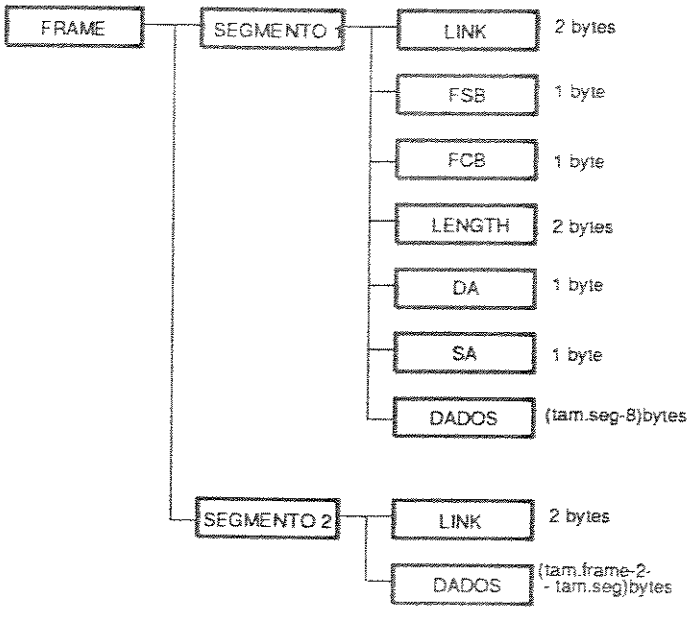

Figura 4.13: Exemplo do Formato de um Frame de 2 Segmentos

No caso de um "frame" de um único segmento, somente o SEGMENTO 1 do exemplo é utilizado; em um "frame" que utiliza mais de 2 segmentos, o formato do SEGMENTO 2 é repetido até que o número total de seus dados sejam armazenados.

RXBUF/WD

Compreende a área da memória "dual-port" Z80/WD, utilizada no armazenamento temporário dos "frames" recebidos pelo controlador de rede local. Assim como o TXBUF/WD, a estrutura do RXBUF/WD é definida para satisfazer o modo de operação do controlador de rede local e, portanto, ela possui o mesmo formato descrito no item anterior.

TAB_LSAP

Este armazenador constitue-se em uma tabela dinâmica que contém informações sobre os SAPs ativados no LLC pela entidade de Gerência da Estação e sobre os Componentes de Transmissão e/ou "Reply-Buffers" criados no decorrer das transações realizadas pelo LLC.

A estrutura proposta para TAB_LSAP pode ser denominada como uma lista de listas, onde as estruturas de dados são interligadas entre si na forma vista na figura 4.14.

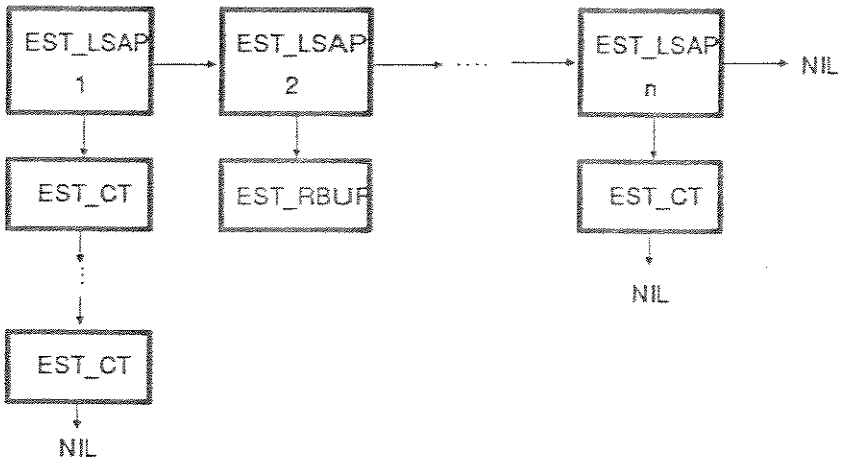

Figura 4.14: Estrutura da TAB_LSAP

Uma estrutura EST_LSAP é criada a cada ativação de um novo LSAP; essa estrutura é preenchida com os parâmetros passados pela primitiva de ativação de LSAPs sendo ligada à

lista de estruturas de dados de outros LSAPs já ativados. A figura 4.15 mostra os elementos que constituem uma estrutura de dados EST_LSAP.

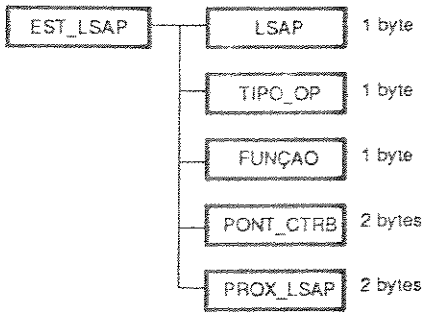

Figura 4.15: Formato da Estrutura de Dados EST_LSAP

Descrição dos campos da estrutura de dados EST_LSAP:

- LSAP: especifica o endereço do LSAP local ativado;
- TIPO_OP: especifica o tipo de operação suportada pelo LSAP (Tipo 1 e/ou Tipo 3);
- FUNÇÃO: especifica a atuação do LSAP nas transações (iniciador, respondedor, iniciador/respondedor ou "reply-buffer");
- PONT_CTRB: contém um ponteiro para o primeiro elemento da lista de estruturas de dados EST_CT ou para uma estrutura EST_RBUF, apresentadas a seguir;
- PROX_LSAP: contém um ponteiro para o próximo elemento da lista de estruturas de dados EST_LSAP.

Uma estrutura EST_CT é criada quando a camada superior envia ao LLC uma primitiva do Tipo 3, para a qual não existe um Componente de Transmissão responsável pelo seu processamento, por esta pertencer a uma transação que envolve uma nova combinação dos parâmetros LSAP local, MSAP remoto e prioridade MAC. Uma vez criada, esta estrutura fornece os meios pelos quais os programas operando na IRL podem identificar e ativar as tarefas referentes aos Componentes de Transmissão. A figura 4.16 mostra os elementos que constituem uma estrutura de dados EST_CT.

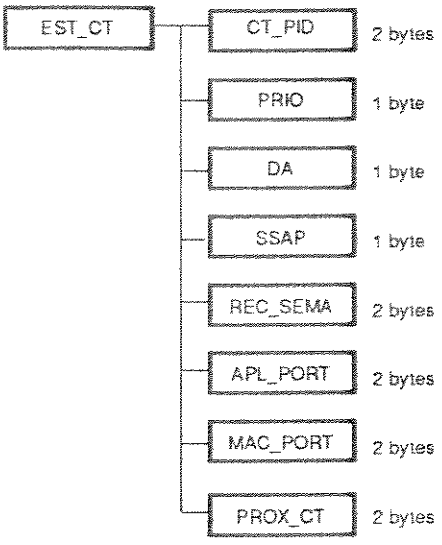

Figura 4.16: Formato da Estrutura de Dados EST_CT

Descrição dos campos da estrutura de dados EST_CT:

- CT_PID: especifica o número que identifica, a nível do NMT, a tarefa do Componente de Transmissão associado a esta estrutura;
- PRIO: prioridade MAC das transações tratadas pelo Componente de Transmissão;
- DA: contém o endereço do MSAP remoto que caracteriza o Componente de Transmissão;
- SSAP: contém o endereço do LSAP local que caracteriza o Componente de Transmissão;
- APL_PORT: especifica a fila do NMT pelo qual o Componente de Transmissão recebe mensagens referentes às transações originadas na camada superior;
- MAC_PORT: especifica a fila do NMT pelo qual o Componente de Transmissão recebe mensagens referentes às transações recebidas da rede local pela subcamada MAC;
- REC_SEMA: especifica o semáforo que acusa o recebimento de uma mensagem em uma das filas citadas acima;
- PROX_CT: contém um ponteiro para o próximo elemento da lista de estruturas de dados EST_CT.

Uma estrutura EST_RBUF é criada juntamente à ativação de um LSAP para a operação Tipo 3 do LLC com a função de ser um "reply-buffer". A figura 4.17 mostra os elementos que constituem uma estrutura EST_RBUF.

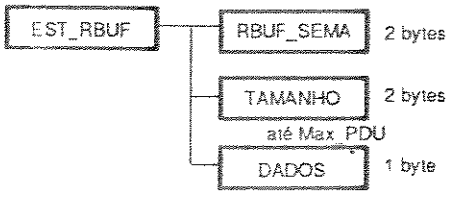

Figura 4.17: Formato da Estrutura de Dados EST_RBUF

Descrição dos campos da estrutura de dados EST_RBUF:

- RBUF_SEMA: especifica o semáforo que realiza a exclusão mútua para acesso ao "reply-buffer";
- TAMANHO: especifica o número de bytes de dados que se encontram armazenados na área DADOS;
- DADOS: contém a unidade de dados armazenada pelo "reply-buffer". A área alocada na criação deste campo deverá ser igual ao comprimento máximo das unidades de dados que ela irá suportar.

TAB_STA

Este armazenador constitue-se em uma tabela dinâmica que contém informações sobre os Componentes de Recepção criados no decorrer das transações gerenciadas pelo LLC. Ele possue a mesma estrutura apresentada na descrição de TAB_LSAP, com a diferença de que nesta tabela têm-se estruturas de dados associadas aos Componentes de Recepção e aos endereços das estações remotas, que participam da caracterização deles. A figura 4.18 mostra a estrutura de TAB_STA.

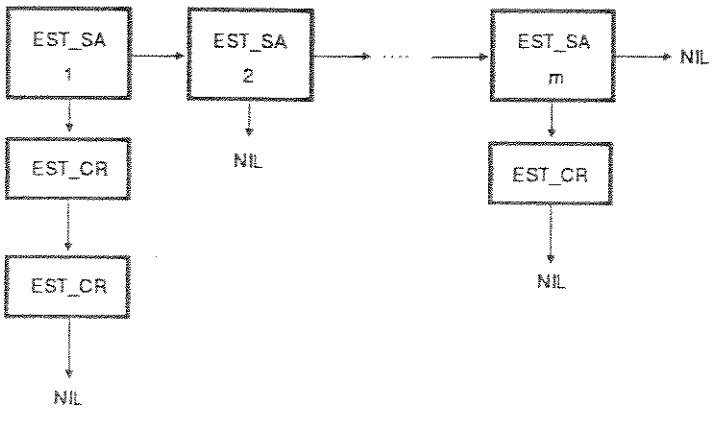

Figura 4.18: Estrutura de TAB_STA

Uma estrutura de dados EST_SA é criada quando o LLC recebe uma PDU Comando do Tipo 3 de uma estação remota, a qual realizou pela primeira vez esse tipo de transação com esta estação. A figura 4.19 mostra os elementos da estrutura de dados EST_SA.

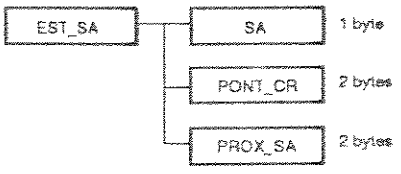

Figura 4.19: Formato da Estrutura de Dados EST_SA

Descrição dos campos da estrutura de dados EST_SA:

- SA: especifica o endereço MSAP da estação remota iniciadora de uma transferência de dados para a estação local;
- PONT_CR: contém um ponteiro para o primeiro elemento da lista de estruturas de dados EST_CR;
- PROX_SA: contém um ponteiro para o próximo elemento da lista de estruturas de dados EST_SA.

Uma estrutura EST_CR é criada quando o LLC recebe uma PDU Comando Tipo 3, para a qual não existe um Componente de Recepção responsável pelo seu processamento, por pertencer a uma transação que envolve uma nova combinação de parâmetros MSAP e LSAP remotos e prioridade MAC. Uma vez criada, essa estrutura fornece os meios pelos quais os programas da IRL podem acessar as tarefas correspondentes aos Componentes de Recepção. A figura 4.20 mostra os elementos da estrutura de dados EST_CR.

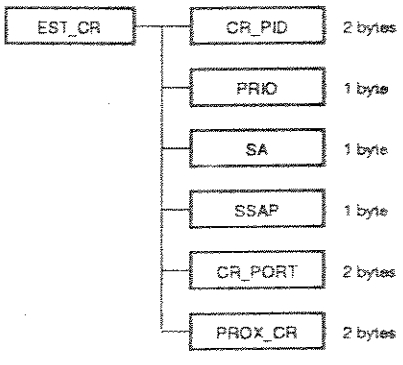

Figura 4.20: Formato da Estrutura de Dados EST_CR

Descrição dos campos da estrutura de dados EST_CR:

- CR_PID: especifica o número que identifica, a nível do NMT, a tarefa do Componente de Recepção associado a esta estrutura;
- PRIO: prioridade MAC das transações tratadas pelo Componente de Recepção;
- SA: contém o endereço do MSAP remoto que caracteriza o Componente de Recepção;
- SSAP: contém o endereço do LSAP remoto que caracteriza o Componente de Recepção;
- CR_PORT: especifica a fila do NMT pelo qual o Componente de Recepção recebe as mensagens referentes às transações originadas na camada superior ou recebidas da rede local pela subcamada MAC;
- PROX_CR: contém um ponteiro para o próximo elemento da lista de estruturas de dados EST_CR.

TTE_CR

Este armazenador de dados constitue-se na Tabela de Transição de Estados do Componente de Recepção. Esta tabela define a máquina de estados finitos que rege a operação de um Componente de Recepção, no gerenciamento das transações de dados sob sua responsabilidade. O armazenador é constituído pelas 9 linhas da tabela, cujos campos são mostrados na figura 4.21.

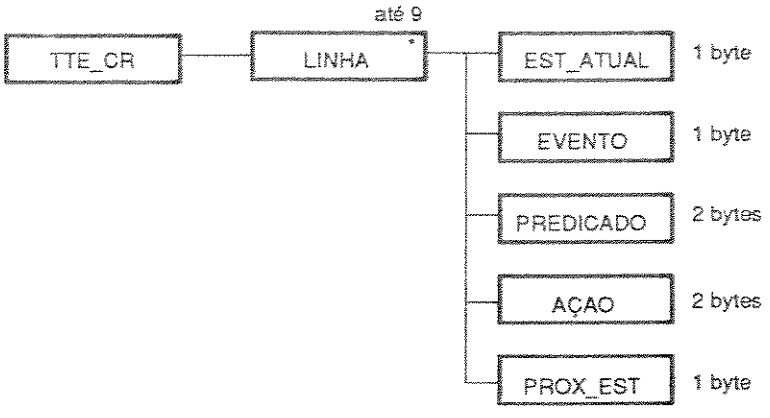

Figura 4.21: Estrutura da TTE_CR

Descrição dos campos da estrutura de dados TTE_CR:

- EST_ATUAL: contém um dos possíveis estados em que a máquina de estados de um Componente de Recepção pode se encontrar;
- EVENTO: contém um dos eventos que excitam a operação da máquina de estados do Componente de Recepção;
- PREDICADO: contém o endereço de uma função C que realiza o teste de predicados referente a uma linha da tabela de transição de estados (as funções de testes de predicados devem retornar um valor booleano, que irá determinar se a função de ação correspondente deverá ser executada ou não);
- AÇÃO: contém o endereço de uma função C que executa uma ação específica da tabela de transição de estados, quando a função de teste de predicados correspondente retorna o valor *verdadeiro*;
- PROX_EST: contém um dos estados no qual um Componente de Recepção poderá ser colocado após a execução de uma dada ação.

TTE_CT

Este armazenador constitue-se na Tabela de Transição de Estados do Componente de Transmissão. Sua estrutura é composta pelos mesmos campos presentes no armazenador TTE_CR, como mostra a figura 4.22, sendo que as duas tabelas diferem somente no número de linhas, que neste caso é de 15 linhas.

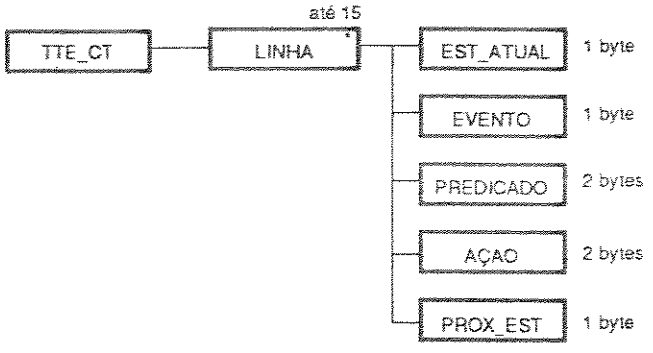

Figura 4.22: Estrutura da TTE_CT

CONF_BUF

Este armazenador é utilizado pela subcamada LLC para guardar informações referentes ao envio de primitivas *MA_DATA.request* à subcamada MAC, permitindo associar primitivas *MA_DATA.confirmation* posteriores aos "requests" correspondetes.

Ele é formado por uma *fila circular* de estruturas de dados que armazenam essas informações. A figura 4.23 mostra os elementos que compõem a estrutura de dados das informações.

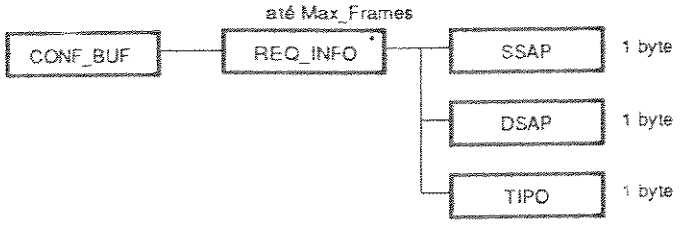

Figura 4.23: Formato da Estrutura de Dados CONF_BUF

Descrição dos campos da estrutura de dados CONF_BUF:

- SSAP: especifica o endereço do LSAP local que originou o "request" feito ao MAC;
- DSAP: especifica o endereço do LSAP remoto para o qual a PDU enviada foi destinada;
- TIPO: contém o tipo de serviço utilizado no envio da PDU (Tipo 1 ou 3) e, no caso de ser do Tipo 3, se é uma PDU de Comando ou de Resposta.

ADD_INFO

Este armazenador guarda os parâmetros utilizados pela tarefa TARSTADD no procedimento de adição de novas estações ao anel lógico da rede local. A estrutura de ADD_INFO é mostrada na figura 4.24.

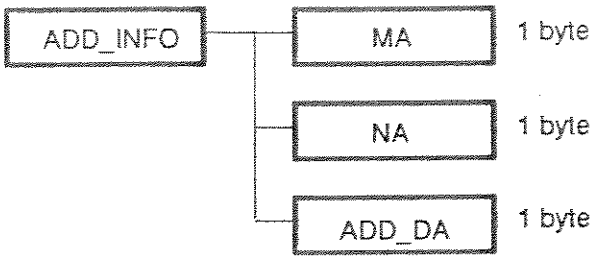

Figura 4.24: Formato da Estrutura de Dados ADD_INFO

Descrição dos campos da estrutura de dados ADD_INFO:

- MA: especifica o endereço físico (MSAP) da estação local;
- NA: especifica o endereço físico da estação remota para a qual a estação local passa o "token", ou seja, a sua estação sucessora no anel lógico da rede;
- ADD_DA: contém o endereço físico de uma suposta estação remota a ser interrogada pela tarefa de adição de novas estações no anel lógico da rede (TARSTADD), para saber se ela existe e se deseja fazer parte dele.

Filas

Os armazenadores GER_PORT, MAC_PORT, APL_PORT, RC_PORT, T1_PORT, IND_PORT e REQ_PORT são as chamadas *filas para comunicação entre tarefas*. Nesta implementação uma fila está sempre relacionada a uma tarefa, ou seja, apenas uma tarefa retira mensagens da sua fila, sendo que várias tarefas podem enviar mensagens para ela. Uma fila é criada quando a tarefa à qual está associada é criada, ambos pelo NMT.

O número de mensagens que cada fila pode suportar é um parâmetro passível de ajustes, como parte do "tunning" do protocolo LLC, em função das necessidades existentes nas várias aplicações da rede local.

4.2.4.2 Descrição das Tarefas

Neste item descreve-se as tarefas implementadas no software da IRL. A figura 4.25 ilustra o *diagrama de controle* [MART86] do sistema, no qual os círculos representam as *tarefas* e as setas representam os *arcos de controle* que permitem visualizar a sequência de ativação das tarefas.

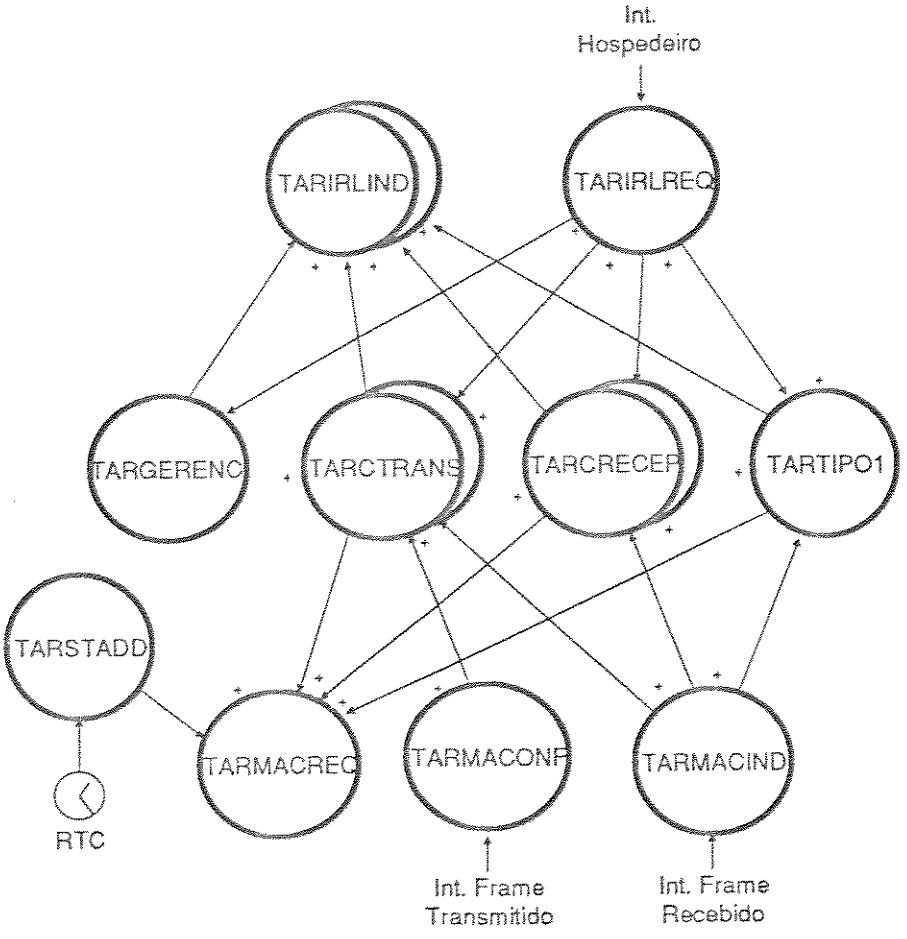

Figura 4.25: Diagrama de Controle do Software da IRL

A seguir cada uma das tarefas são apresentadas e seus procedimentos descritos na forma de pseudo-código.

TARIRLREQ

Esta tarefa é responsável, basicamente, pela identificação do serviço requisitado pelo hospedeiro, através da primitiva por ele enviada, e pelo seu encaminhamento à tarefa que efetivamente irá tratá-la. Ela é ativada pela rotina de tratamento da interrupção gerada pelo hospedeiro, através da sinalização do semáforo contador do número de primitivas recebidas, HSTREQSEMA. À cada ativação, TARIRLREQ inicia a busca de uma primitiva recebida no TXBUF/HOST a partir da sub-área de maior prioridade, garantindo o tratamento das primitivas pela ordem de suas prioridades e não pela ordem de envio do hospedeiro. A sincronização para que as primitivas de mesma prioridade sejam retiradas do TXBUF/HOST de forma sequencial, é feita utilizando o semáforo associado à cada sub-área desse "buffer"; esses semáforos fazem a exclusão mútua das tarefas que tratam as primitivas de uma mesma

prioridade, não permitindo que TARIRLREQ ative uma nova tarefa, enquanto uma outra ainda não terminou o tratamento de uma primitiva anterior.

Abaixo são mostradas, resumidamente, as operações executadas por TARIRLREQ, na forma de pseudo-código.

```
TARIRLREQ( )
begin
   while( IRL operacional )
      Espera sinalização do semáforo HSTREQSEMA;
      Procura no TXBUF/BUS uma primitiva de maior prioridade;
      TrataPrimitiva( );
   endwhile
end


TrataPrimitiva( )
begin
   Obtém o campo primitiva contido na mensagem;
   case ( primitiva ) of
      L_RESET_REQ:                ReqGerência( );
      L_STATUS_REQ:               ReqGerência( );
      L_SAP_ACTIVATE_REQ:         ReqGerência( );
      L_SAP_DEACTIVATE_REQ:       ReqGerência( );
      L_RSAP_ACTIVATE_REQ:        ReqGerência( );
      M_INIT_PROTOCOL_REQ:        ReqGerência( );
      M_SET_VALUE_REQ:            ReqGerência( );
      M_READ_VALUE_REQ:           ReqGerência( );
      M_CDATA_REQ:                ReqGerência( );
      M_GROUP_ADDRESS_REQ:        ReqGerência( );
      DL_DATA_ACK_REQ:            ProcuraCT( );
      DL_REPLY_REQ:               ProcuraCT( );
      DL_REPLY_UPDATE_REQ:        ProcuraCR( );
      L_DATA_REQ:                 ServTipo1( );
   endcase
end
```

```
ReqGerência( )
begin
   Cria mensagem do NMT com dados da primitiva;
   Envia mensagem para a fila da tarefa TARGERENC;
end


ServTipo1( )
begin
   Cria mensagem do NMT com dados da primitiva;
   Envia mensagem para a fila da tarefa TARTIPO1;
end


ProcuraCT( )
begin
   Procura em TAB_LSAP uma estrutura EST_CT caracterizada pelos  parâmetros da transação;
   if ( não existe a estrutura EST_CT )
      Cria uma instância da tarefa TARCTRANS para tratá-la;
      Coloca em TAB_LSAP a estrutura EST_CT criada;
   endif
   Cria mensagem do NMT com dados da primitiva;
   Envia mensagem para a fila da instância da tarefa TARCTRANS associada à  EST_CT;
end


ProcuraCR( )
begin
   Procura em TAB_STA uma estrutura EST_CR qualquer ;
   if ( existe pelo menos uma estrutura EST_CR )
      Cria uma mensagem do NMT com dados da primitiva;
      Envia mensagem para a fila da instância da tarefa TARCRECEP associada à EST_CR;
   else
      Retira primitiva de TXBUF/BUS sem executá-la;
   endif
end
```

TARGERENC

Esta tarefa é responsável pelo tratamento das primitivas enviadas pela entidade de Gerência da Estação às subcamadas LLC e MAC. Ela possue apenas uma instância pelo fato de tratar serviços de abrangência local à estação e que, na maioria dos casos, são utilizados na iniciação do nó de rede, ou seja, em procedimentos sequenciais (a Gerência aguarda a confirmação da execução de uma primitiva antes de enviar a seguinte). Sua ativação ocorre

através da recepção de mensagens, a nível do NMT, vindas da tarefa TARIRLREQ; a fila GER_PORT é utilizada para receber as mensagens geradas por TARIRLREQ.

As funções contidas nesta tarefa devem, basicamente, identificar os serviços requisitados, executá-los e, confirmar o sucesso ou não da sua execução, retornando algum valor quando solicitado.

Nesta primeira fase do projeto foram implementadas apenas as primitivas necessárias para tornar as subcamadas LLC e MAC operacionais. Na apresentação dos pseudo-códigos, a seguir, as primitivas não implementadas serão apenas comentadas, enquanto que as implementadas terão seus pseudo-códigos mostrados.

```
TARGERENC( )
begin
   while( IRL operacional )
      Espera por uma mensagem na fila GER_PORT;
      Obtém o campo primitiva contido na mensagem;
      case( primitiva ) of
         L_RESET_REQ:                ResetaLLC( );
         L_STATUS_REQ:               SapStatus( );
         L_SAP_ACTIVATE_REQ:         AtivaSap( );
         L_SAP_DEACTIVATE_REQ:       DesatSap( );
         L_RSAP_ACTIVATE_REQ:        AtivaRsap( );
         M_INITIALIZE_PROTOCOL_REQ:  InicProt( );
         M_SET_VALUE_REQ:            SetaValor( );
         M_READ_VALUE_REQ:           LeValor( );
         M_CDATA_REQ:                DadosGer( );
         M_GROUP_ADDRESS_REQ:        GroupAddr( );
      endcase
   Atualiza ponteiro RETIRA da sub-área do TXBUF/BUS, associada à mensagem tratada;
   endwhile
end
```

```
ResetaLLC( )
/*
   Esta função retorna o LLC no seu estado inicial, isto é, basicamente o conteúdo das tabelas
   TAB_LSAP e TAB_STA é eliminado e as áreas de memória ocupadas pelas estruturas de
   dados que compunham essas tabelas são retornadas ao NMT. Além disso, as tarefas
   associadas aos Componentes de Transmissão e Componentes de Recepção existentes
   devem ser "mortas".
*/
```

```
SapStatus( )
/*
   Esta função verifica se um certo LSAP se encontra ativado e o tipo de operação que ele
   executa.
*/


AtivaSap( )
begin
   Obtém o campo LSAP local contido na mensagem;
   Procura LSAP local em TAB_LSAP;
   if( TAB_LSAP está vazia ou ela não possui o LSAP )
      Cria estrutura associada ao LSAP (EST_LSAP) e a coloca em TAB_LSAP;
   endif
   Confirma o sucesso ou não dessa operação à Gerência da Estação;
end


DesatSap( )
/*
   Executa a desativação de um único LSAP específico, ou seja, sua estrutura de dados é
   retirada de TAB_LSAP e retornada ao NMT. Além disso, quaisquer Componentes de
   Transmissão associados a este LSAP devem ser destruídos, isto é, suas estruturas de dados
   presentes em TAB_LSAP retornadas ao NMT e suas tarefas "mortas".
*/


AtivaRsap( )
/*
   Esta função ativa um LSAP que irá atuar como um "reply-buffer". O procedimento é
   semelhante ao descrito na função AtivaSap, com a adição de uma área de memória que é
   reservada para armazenar os dados passados pelas primitivas DL_REPLY_UPDATE request.
*/


InicProt( )
/*
   Retira o MAC do anel lógico da rede local, através da programação do "chip" controlador de
   rede.
*/
```

```
SetaValor( )
begin
   Obtém o nome da variável a ser programada contido na mensagem;
   case( variável ) of
      IN_RING:        Network( );      /* faz com que o nó tente entrar no anel lógico */
      THIS_STA:       SetMA( );        /* programa endereço físico do nó */
      SLOT_TIME:      Slotime( );      /* programa valor do parâmentro SLOT_TIME */
   endcase
   Confirma o sucesso ou não dessa operação à Gerência da Estação;
end


Network( )
begin
   Programa o WD2840 para entrar no modo de operação NETWORK;
end


SetMA( )
begin
   Obtém o campo do endereço físico a ser dado à estação, contido na mensagem;
   if( endereço inválido )
      Retorna à função SetaValor, indicando erro;
   endif
   Atualiza a estrutura de dados WDINFO com o novo valor;
   Programa o WD2840 para assumir o novo endereço na rede local;
end


Slotime( )
begin
   Obtém o valor para SLOT_TIME contido na mensagem;
   if( valor é inválido )
      Retorna à função SetaValor, indicando erro;
   endif
   Programa registro TA do WD2840 com esse valor;
end
```

```
LeValor( )
begin
   Obtém o nome da variável a ser lida, contido na mensagem;
   case( variável ) of
      IN_RING:          Inring( );          /* verifica se o nó já entrou no anel lógico */
   endcase                                  /* outras variáveis serão usadas futuramente */
   Confirma à Gerência da Estação, retornando o valor lido;
end


Inring( )
begin
   Retorna valor do bit INRING do reg. SR0 do WD2840;
end


DadosGer( )
/*
   A transferência de dados entre as entidades de Gerência da Estação não faz parte do
   escopo deste trabalho.
*/


GroupAddr( )
/*
   Não aplicável neste projeto uma vez que o "chip" controlador de rede não suporta
   endereçamento de grupo (apenas individual ou de "broadcast").
*/
```

TARCTRANS

A função desta tarefa é a de implementar um Componente de Transmissão para tratar as primitivas *DL_DATA_ACK request* e *DL_REPLY request*, enviadas pela camada de Aplicação, e as PDUs de Resposta enviadas por outras subcamadas LLC. Como visto anteriormente, o LLC de uma dada estação possuirá um Componente de Transmissão para tratar cada possível combinação dos parâmetros LSAP local, MSAP remoto e prioridade MAC, referentes a uma certa transação. Uma vez que um Componente de Transmissão somente trata uma transação por vez e as mensagens a serem transmitidas podem ter prioridades distintas, optou-se pela implementação de várias instâncias da tarefa TARCTRANS, sendo cada instância associada a um Componente de Transmissão. A criação de uma instância de TARCTRANS é feita dinamicamente pela tarefa TARIRLREQ, ao receber uma primitiva para a qual não existe um Componente de Transmissão associado. A prioridade atribuída à instância criada , a nivel do NMT, é dada em função da prioridade MAC das transações que este Componente de Transmissão irá tratar.

A ativação de uma instância de TARCTRANS ocorre através da recepção de mensagens, a nível do NMT, vindas das tarefas TARIRLREQ, TARMACIND e TARMACONF. Dependendo do estado em que se encontra o Componente de Transmissão associado à instância ativada, o evento recebido nessas mensagens causará a execução da máquina de estados finitos, segundo a Tabela de Transição de Estados do Componente de Transmissão (TTE_CT).

O pseudo-código das operações realizadas por TARCTRANS é mostrado abaixo.

```
TARCTRANS( )
begin
   Inicia parâmetros internos do CR;
   while( LLC ativo )
      Espera sinalização do semáforo de recepção de mensagens;
      if( existe mensagem na fila MAC_PORT (vinda do MAC) )
         Recebe mensagem de MAC_PORT;
         Trata-a segundo a Tabela de Transição de Estados do CT;
      else
         if( estado atual do CT for IDLE )
            Recebe mensagem da fila APL_PORT (vinda da camada de Aplicação);
            Trata-a segundo a Tabela de Transição de Estados do CT;
         else
            Incrementa contador das mensagens em APL_PORT;
         endif
      endif
      if( contador das mensagens em APL_PORT  0 e estado atual do CT for IDLE )
         Recebe mensagem da fila APL_PORT;
         Trata-a segundo a Tabela de Transição de Estados do CT;
      endif
   endwhile
end
```

TARCRECEP

A função desta tarefa é a de implementar um Componente de Recepção para tratar as PDUs de Comando, recebidas de outras subcamadas LLC, e a primitiva *DL_REPLY_UPDATE request*, vinda do hospedeiro. Como visto anteriormente, o LLC de uma dada estação possuirá um Componente de Recepção para tratar cada possível combinação dos parâmetros LSAP remoto, MSAP remoto e prioridade MAC, referentes a uma certa transação. Devido a essa prioridade, optou-se pela implementação de várias instâncias da tarefa TARCRECEP, sendo cada uma delas associada a um Componente de Recepção, como no caso da tarefa TARCTRANS. A criação de uma instância de

TARCRECEP é feita dinamicamente pela tarefa TARMACIND, ao receber uma PDU de Comando, para a qual não existe um Componente de Recepção associado. A prioridade atribuída à instância criada , a nivel do NMT, é dada em função da prioridade MAC referente às transações tratadas pelo Componente de Recepção.

A ativação de uma instância de TARCRECEP ocorre através da recepção de mensagens, a nível do NMT, vindas das tarefas TARIRLREQ e TARMACIND. O evento contido nessas mensagens causa a execução da máquina de estados finitos, segundo a Tabela de Transição de Estados do Componente de Recepção.

O pseudo-código das operações realizadas por TARCRECEP é mostrado abaixo.

```
TARCRECEP( )
begin
    Inicia parâmetros internos do CR;
    while( LLC ativo )
        Recebe uma mensagem da fila RC_PORT;
        Trata-a segundo a Tabela de Transição de Estados do CR;
    endwhile
end
```

TARTIPO1

Esta tarefa tem a responsabilidade de tratar as transações referentes à operação Tipo 1 do LLC. Por se tratar de uma tarefa de pouca complexidade e seu uso restrito apenas às transações envolvendo LSAPs com endereçamento de grupo ou "broadcast" [GEMO88], ela não foi implementada nesta versão do projeto.

TARMACIND

Esta tarefa é responsável, primeiramente, em identificar o tipo do "frame" recebido pela subcamada MAC e, posteriormente, encaminhar o seu tratamento à função ou tarefa pertinente. Como visto em 4.2.3, os dois tipos de "frames" implementados são: *frame de controle do MAC* e *frame de dados do LLC*. No caso do primeiro tipo, seu tratamento é imediato e é resolvido por uma função interna a esta tarefa; no segundo tipo de "frame", é necessário que o LLC identifique o tipo da PDU recebida para, então, enviá-la à tarefa responsável em tratá-la.

A ativação desta tarefa é feita pela rotina de tratamento das interrupções geradas pelo "chip" controlador de rede local. Como esta interrupção se deve à chegada de um "frame" vindo da rede local, a rotina sinaliza um semáforo contador do número de "frames" recebidos, RECVSEM. Como o "chip" controlador de rede local trabalha com um único "buffer" para recepção, a cada ativação a tarefa faz uma busca em RXBUF/WD usando um esquema "FIFO", não sendo levado em conta a prioridade das transações nesse instante; a prioridade

somente passa a influenciar no tratamento das transações a partir do momento em que TARMACIND ativa as tarefas subsequentes.

Apesar dos "frames" serem colocados no RXBUF/WD pela ordem de chegada, a estrutura desse "buffer" não exige que o tratamento dos "frames" seja feito na mesma ordem. TARMACIND pode passar os "frames" recebidos para as tarefas que irão tratá-los sem a necessidade de esperar pelo término de qualquer uma delas, possibilitando, assim, um certo grau de paralelismo no processamento. É preciso somente que haja uma exclusão mútua entre essas tarefas no momento em que, após finalizarem o processamento de um "frame", elas forem liberar o(s) segmento(s) utilizado(s) pelo "frame", o que implica em alterar ponteiros que gerenciam o RXBUF/WD.

As operações realizadas por TARMACIND são mostradas a seguir, na forma de um pseudo-código.

```
TARMACIND( )
begin
   while( LLC ativo )
      Espera sinalização do semáforo RECVSEM;
      Obtém o byte de controle do "frame" recebido;
      case ( byte de controle do frame ) of
         frame_de_dados_do_LLC:        MaDataInd( );
         frame_de_controle_do_MAC:     MacFrame( );
      endcase
   endwhile
end
```

```
MaDataInd( )
begin
   case ( byte de controle da PDU_LLC ) of
      comando_UI:     Tip1Pdu( );
      comando_XID:    Tip1Pdu( );
      comando_TEST:   Tip1Pdu( );
      comando_ACn:    Tip3Pdu( );
      resposta_ACn:   Tip3Pdu( );
   endcase
end
```

```
Tip1Pdu( )
begin
   if ( LSAP destino da PDU está ativado )
      Envia mensagem para tarefa TARTIPO1;
   else
      Deleta "frame" do RXBUF/WD;
   endif
end


Tip3Pdu( )
begin
   if ( bit C/R do campo SSAP da PDU indica comando )
      CmdPdu( );
   else
      RspPdu( );
   endif
end

CmdPdu( )
begin
   if ( LSAP destino da PDU está ativado )
      Procura em TAB_STA uma estrutura EST_CR caracterizada pelos parâmetros da transação;
      if ( não existe a estrutura EST_CR )
         Cria uma instância da tarefa TARCRECEP para tratá-la;
         Coloca em TAB_STA a estrutura EST_CR criada;
      endif
      Envia mensagem para a fila da instância da tarefa TARCRECEP associada;
      Retorna à função Tip3Pdu;
   else
      Deleta "frame" recebido no RXBUF/WD;
   endif
end

RspPdu( )
begin
   Procura em TAB_LSAP uma estrutura EST_CT caracterizada pelos parâmetros da transação;
   if ( existe a estrutura EST_CT )
      Envia mensagem para a fila da instância da tarefa TARCTRANS associada;
      Retorna à função Tip3Pdu:
   else
      Deleta "frame" recebido no RXBUF/WD;
   endif
end
```

```
MacFrame( )
begin
   Deleta "frame" recebido no RXBUF/WD ( já tratado pelo WD2840);
end
```

TARMACREQ

Apesar de aparecer no diagrama de controle GMB como sendo uma tarefa, TARMACREQ é na realidade uma função que é chamada internamente às tarefas TARTIPO1, TARCTRANS, TARCRECEP e TARSTADD, utilizada na colocação de "frames" em TXBUF/WD para envio à rede local. Esta função implementa a primitiva *MA_DATA.request*, utilizada na passagem de informações da subcamada LLC para a subcamada MAC. TARMACREQ é apresentada como uma única tarefa para indicar que o acesso à TXBUF/WD tem que ser feito através de exclusão mútua entre as tarefas que a chamam, devido ao fato do controlador de rede também operar com um "buffer" único para transmissão. Ela poderia ser implementada como sendo uma tarefa, entretanto, para se evitar o "overhead" causado pelo procedimento de troca de mensagens do NMT, optou-se em fazer essa exclusão mútua utilizando-se um semáforo WDTXSEMA, o qual é testado pelas tarefas que desejam colocar "frames" em TXBUF/WD, antes de iniciar a colocação efetivamente.

As funções contidas em TARMACREQ são mostradas a seguir, na forma de pseudo-código.

```
TARMACREQ( )
begin
   Calcula o tamanho do "frame" a ser gerado;
   Faz exclusão mútua de acesso à TXBUF/WD (semáforo WDTXSEMA);
   Verifica se existe espaço em TXBUF/WD para o "frame";
   while ( não existe espaço em TXBUF/WD )
      Coloca tarefa chamadora em SLEEP por um período de tempo;
   endwhile
   Monta o "frame" em TXBUF/WD;
   Atualiza ponteiro COLOCA do TXBUF/WD para próximo "frame";
   Termina exclusão mútua de acesso à TXBUF/WD;
end
```

TARIRLIND

Assim como a tarefa descrita anteriormente, TARIRLIND aparece no diagrama de controle GMB representada como uma tarefa. Entretanto, ela constitue-se em uma função chamada internamente às tarefas TARGERENC, TARTIPO1, TARCTRANS e TARCRECEP, utilizada na colocação de primitivas em RXBUF/HOST, para envio ao

hospedeiro. TARIRLIND é apresentada como tendo várias instâncias, na realidade 8 (uma para cada prioridade MAC), para indicar que a cada instância está associado o gerenciamento de uma sub-área de RXBUF/HOST; isso permite que a colocação de primitivas de prioridades diferentes possa ser feita concorrentemente. Na prática, a concorrência para a colocação das primitivas é feita em função da prioridade a nível do NMT associada às tarefas que utilizam as funções contidas em TARIRLIND. Da mesma forma que na tarefa TARIRLREQ, é necessário se garantir a colocação de primitivas de forma sequencial em RXBUF/HOST, para uma mesma sub-área (mesma prioridade MAC); isto é obtido com a utilização de um semáforo para cada uma delas, o qual realiza a exclusão mútua das tarefas que desejam colocar primitivas em uma mesma sub-área.

As funções contidas em TARIRLIND são mostradas a seguir, na forma de pseudo-código.

```
TARIRLIND( )
begin
   Obtém a prioridade MAC da transação;
   Faz exclusão mútua de acesso à sub-área de RXBUF/HOST associada à prioridade;
   Calcula o tamanho da primitiva a ser gerada;
   Verifica se existe espaço nessa sub-área de RXBUF/HOST para a primitiva;
   while ( não existe espaço em RXBUF/HOST )
      Coloca tarefa chamadora em SLEEP por um período de tempo;
   endwhile
   Monta a primitiva em RXBUF/HOST;
   Atualiza ponteiro COLOCA do RXBUF/HOST;
   Termina exclusão mútua de acesso à sub-área de RXBUF/HOST;
   Gera interrupção no hospedeiro;
end
```

TARMACONF

Esta tarefa tem a função básica de confirmar ao LLC o sucesso ou não no envio dos *"frames" de dados do LLC*, por ele requisitado. Porém, ela também é utilizada no procedimento de adição de novas estações no anel lógico da rede local, interpretando o resultado da transmissão dos *"frames" de controle do MAC*.

A ativação desta tarefa é feita pela rotina de tratamento das interrupções geradas pelo "chip" controlador de rede local. Como a interrupção associada a esta tarefa ocorre a cada transmissão dos "frames" contidos em TXBUF/WD, quando do recebimento do "token" da rede local, a rotina de interrupção sinaliza um semáforo contador do número de transmissões realizadas (CONFSEMA).

A cada ativação de TARMACONF são tratados todos os "frames" transmitidos pelo controlador de rede, até aquele instante. Após o tratamento de cada "frame", a área de

TXBUF/WD ocupada por ele é liberada para ser re-utilizada posteriormente; essa operação resulta na alteração de ponteiros de gerenciamento do TXBUF/WD e, portanto, necessita de exclusão mútua, que é realizada através do semáforo WDTXSEMA associado a esse "buffer".

As operações realizadas por TARMACONF são mostradas a seguir, na forma de pseudo-código.

```
TARMACONF( )
begin
   while( LLC ativo )
      Espera sinalização do semáforo CONFSEMA;
      Faz exclusão mútua de acesso à TXBUF/WD (semáforo WDTXSEMA);
      while( existem "frames" já transmitidos em TXBUF/WD )
         Obtém campo controle_do_frame do "frame" sendo tratado;
         case( controle_do_frame ) of
            frame_de_dados_do_LLC:     MaDataConf( );
            frame_de_controle_do_MAC:  MacControl( );
         endcase
         Deleta "frame" confirmado no TXBUF/WD;
      endwhile
   endwhile
end

MaDataConf( )
begin
   Obtém estrutura de dados apontada pelo ponteiro RETIRA em CONF_BUF;
   Verifica o tipo da PDU recebida;
   if( PDU sendo tratada é do Tipo 3 e é de Comando )
      Procura em TAB_LSAP uma estrutura EST_CT caracterizada pelos parâmetros da
      transação;
      Envia mensagem para a fila da instância da tarefa TARCTRANS associada;
   endif
   Atualiza ponteiro RETIRA do CONF_BUF;
end

MacControl( )
begin
   if( "frame" tratado foi de "polling" de estação )
      if( estação destino deseja entrar na rede local )
         Atualiza estrutura ADD_INFO e o registro NA do WD2840 com endereço da nova estação;
      endif
      Incrementa campo ADD_DA de ADD_INFO com o endereço da próxima estação a ser
      interrogada;
   endif
end
```

TARSTADD

Esta tarefa é responsável pela adição de novas estações ao anel lógico da rede local, cujos endereços estão compreendidos entre o endereço da estação que inicia este procedimento e o de sua sucessora. A implementação desta tarefa se faz necessária uma vez que o "chip" utilizado neste projeto não possui o procedimento de adição de estações especificado pela norma IEEE 802.4.

A ativação desta tarefa é feita em espaços de tempos pré-determinados, através do uso de uma função do NMT, que temporiza o período de dormência da tarefa [COME84]. A cada ativação da tarefa, uma estação é interrogada para se saber se ela deseja entrar no anel lógico.

As operações realizadas por TARSTADD são mostradas a seguir, na forma de pseudo-código.

```
TARSTADD( )
begin
   while( estação se acha no anel lógico da rede )
      Espera esgotar o tempo de "sleep";
      if ( existem endereços intermediários entre esta estação e sua sucessora )
         Envia uma mensagem a um desses endereços, para saber se a estação deseja entrar na
         rede local;
      endif
   endwhile
end
```

4.2.5 Exemplo da Dinâmica de Operação do Software

Este item visa exemplificar de uma forma simplificada a dinâmica dos programas presentes na IRL, em função de eventos das interfaces entre as camadas. Para salientar os aspectos envolvidos na criação das estruturas de dados, os exemplos irão se basear na operação da subcamada LLC, uma vez que os programas relativos à subcamada MAC não apresentam complexidades quanto à dinâmica das estruturas de dados.

O exemplo irá abranger duas estações realizando uma sequência de eventos, a começar pela ativação de LSAPs, para em seguida ser executada uma transferência de dados, bem sucedida, entre elas, utilizando-se o serviço Tipo 3 do LLC, *DL_DATA_ACK request*.

Ativação dos L_SAPs envolvidos na transferência de dados

A figura 4.26 mostra o envio da primitiva *L_SAP_ACTIVATE.request* pela Gerência da Estação, solicitando ao LLC que um dado LSAP seja ativado. Para simplificação do exemplo, a figura mostra a mesma primitiva sendo enviada às duas estações coincidentemente. Pode-se notar ainda que as estruturas de dados presentes na figura encontram-se vazias.

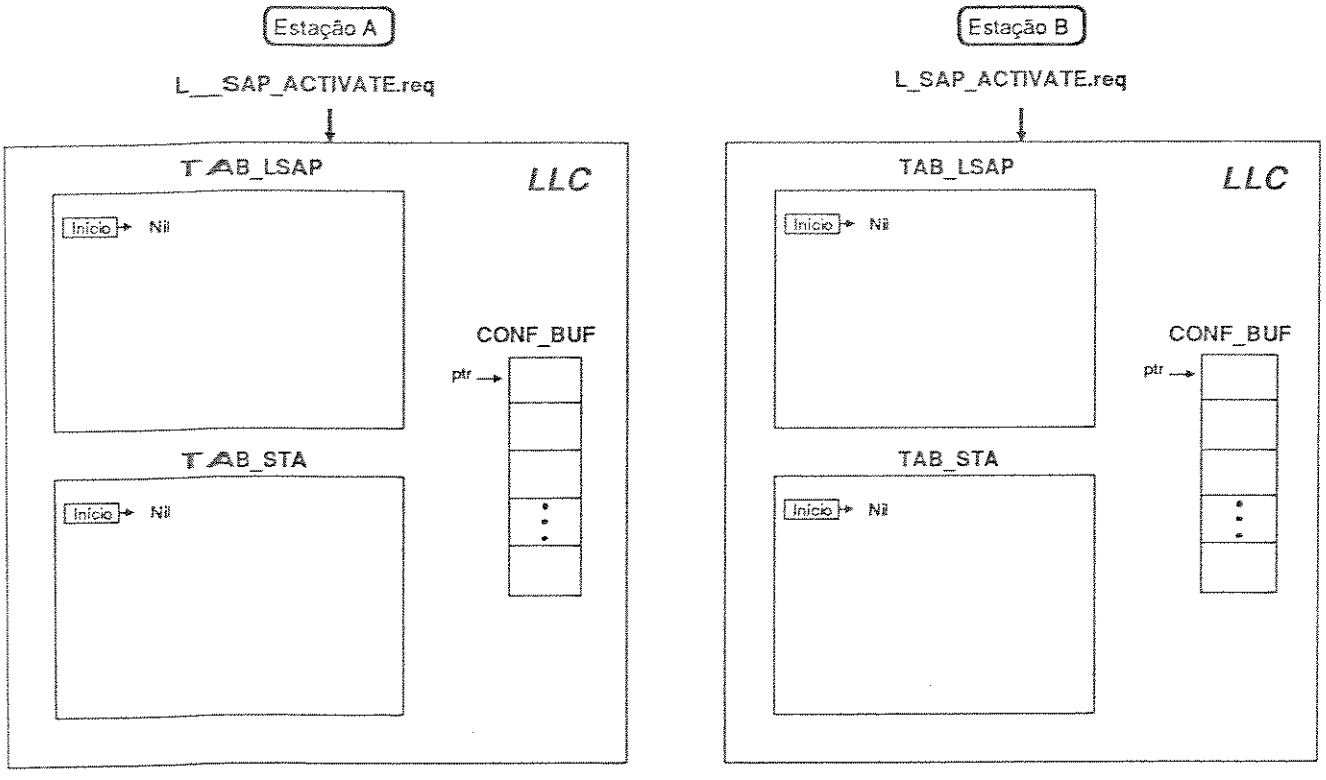

Figura 4.26: Pedido de Ativação de um LSAP

A chegada desta primitiva no LLC ativa a tarefa TARGERENC que cria uma estrutura EST_LSAP e coloca-a em TAB_LSAP. Em seguida, TARGERENC gera uma primitiva *L_SAP_ACTIVATE.confirm* para a Gerência da Estação, encerrando a execução do pedido, como mostra a figura 4.27.

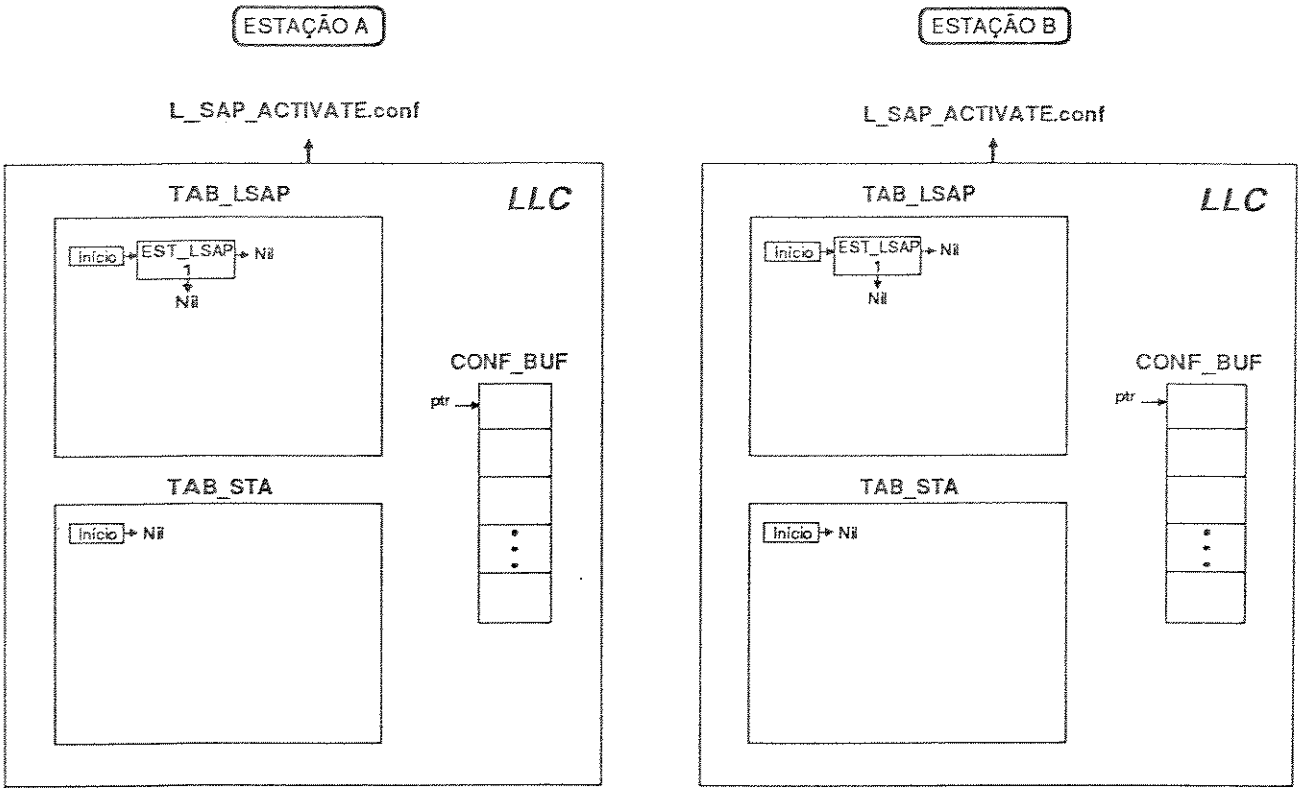

Figura 4.27: Confirmação da Ativação de um LSAP

Transferência de dados entre as estações

Na figura 4.28 a camada de Aplicação da estação A requisita ao LLC que uma unidade de dados seja enviada à estação B, através de um serviço sem conexão e com confirmação, ou seja, usando-se a primitiva *DL_DATA_ACK request*. Os LSAPs envolvidos na transação são os mesmos ativados anteriormente.

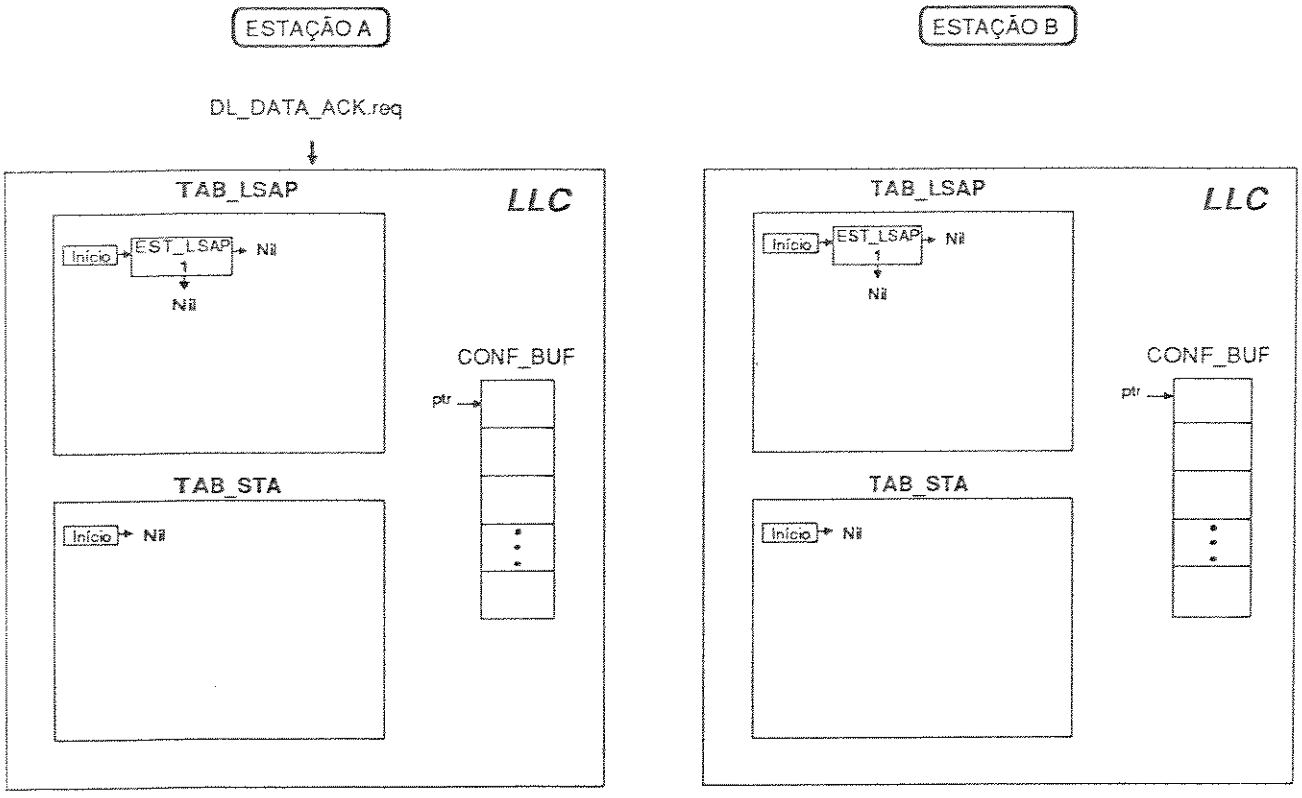

Figura 4.28: Pedido de Envio de Dados para o LLC_A

Este evento faz com que o LLC crie uma estrutura EST_CT e coloque-a em TAB_LSAP. Em seguida, uma instância da tarefa TARCTRANS é criada e ativada.

Esta instância, por sua vez, realizará o tratamento da primitiva, executando a máquina de estados do Componente de Transmissão que resultará no envio de uma primitiva *MA_DATA.request* para o MAC (figura 4.29). Este se encarregará de enviar um "frame" de dados para a estação B através da rede local. Ao solicitar este serviço ao MAC, o LLC armazena em CONF_BUF informações referentes à transação sendo feita, para que ao receber uma primitiva *MA_DATA.confirmation*, ela possa ser associada ao pedido realizado anteriormente.

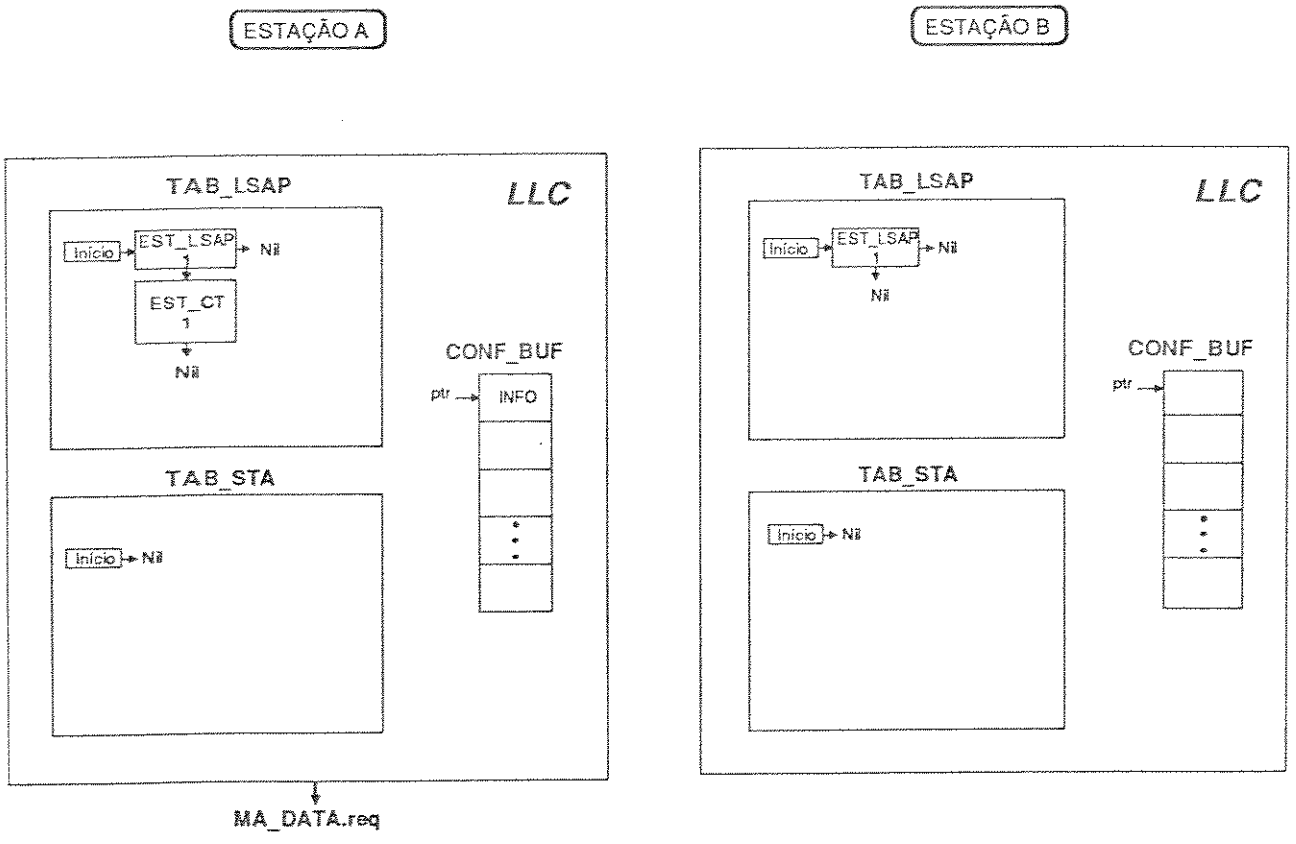

Figura 4.29: LLC_A Pede ao MAC_A para Enviar Dados

Novamente para simplificar o exemplo, a figura 4.30 mostra o LLC da estação A recebendo uma primitiva *MA_DATA.confirmation*, enquanto que o LLC da estação B recebe uma primitiva *MA_DATA.indication*.

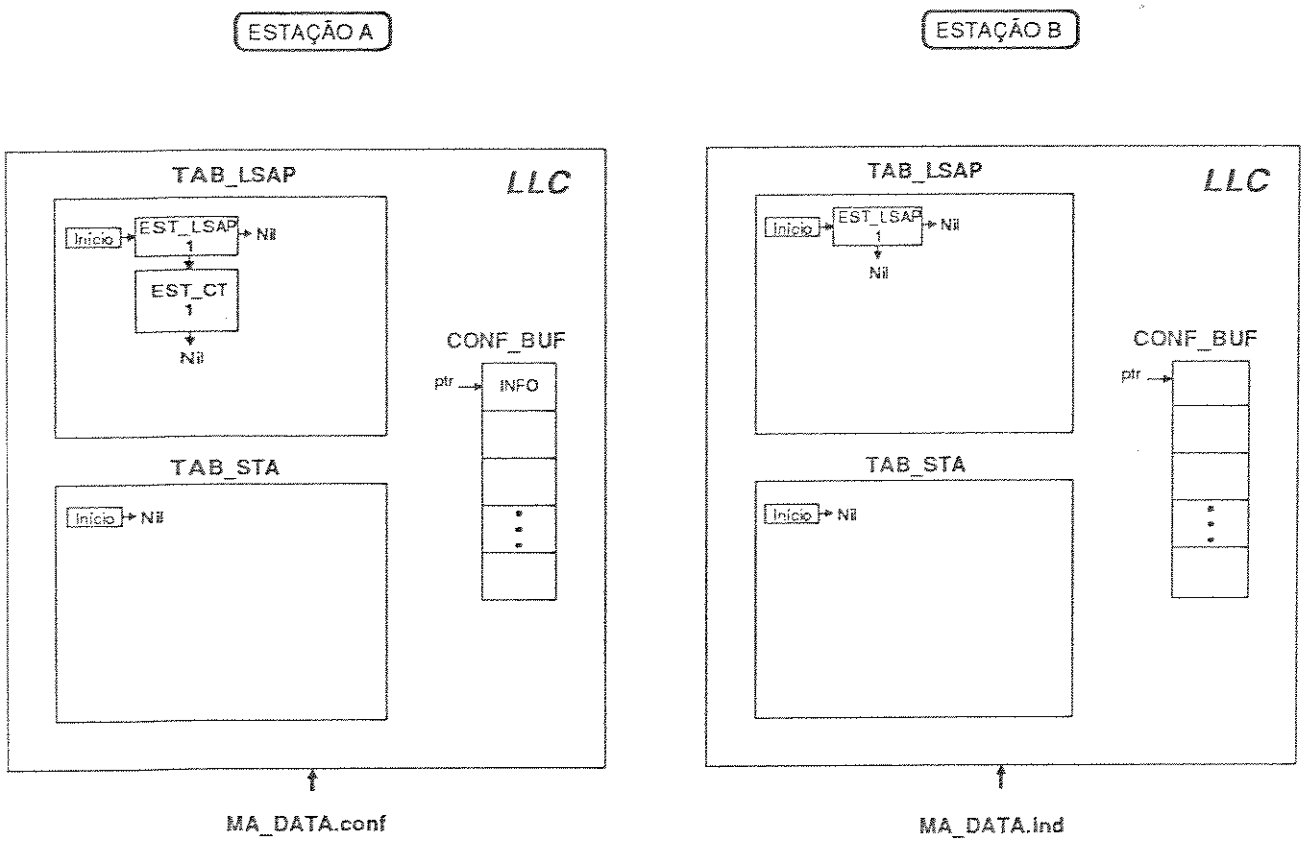

Figura 4.30: LLC_A e LLC_B Recebem Primitivas de seus MACs

A primitiva da estação B é gerada pela chegada do "frame" vindo da estação A e a primitiva da estação A é gerada para indicar o sucesso ou falha da transmissão do "frame" pela estação A. Esses dois eventos de interface ocasionam, nessas estações, as ações descritas abaixo (ver figura 4.31):

- *na estação A*: ao receber a primitiva o LLC obtém em CONF_BUF as informações da transação desenvolvida, a fim de endereçar o Componente de Transmissão e ativar sua máquina de estados, alterando seu estado em função do "status" da transmissão feita;
- *na estação B*: o LLC desta estação obtém os parâmetros SA, SSAP e PRIORIDADE da primitiva recebida e procura em TAB_STA se existe um Componente de Recepção caracterizado por esses valores; como TAB_STA encontra-se vazia, uma estrutura EST_SA e uma EST_CR são criadas e colocadas em TAB_STA. Além disso, uma instância da tarefa TARCRECEP é criada e ativada, e sua máquina de estados realiza o tratamento deste evento, enviando a unidade de dados originada pela camada de Aplicação da estação A, para a da estação B, na forma de uma primitiva *DL_DATA_ACK.indication* e, também, enviando uma resposta ao LLC da estação A, através da primitiva *MA_DATA.request* .

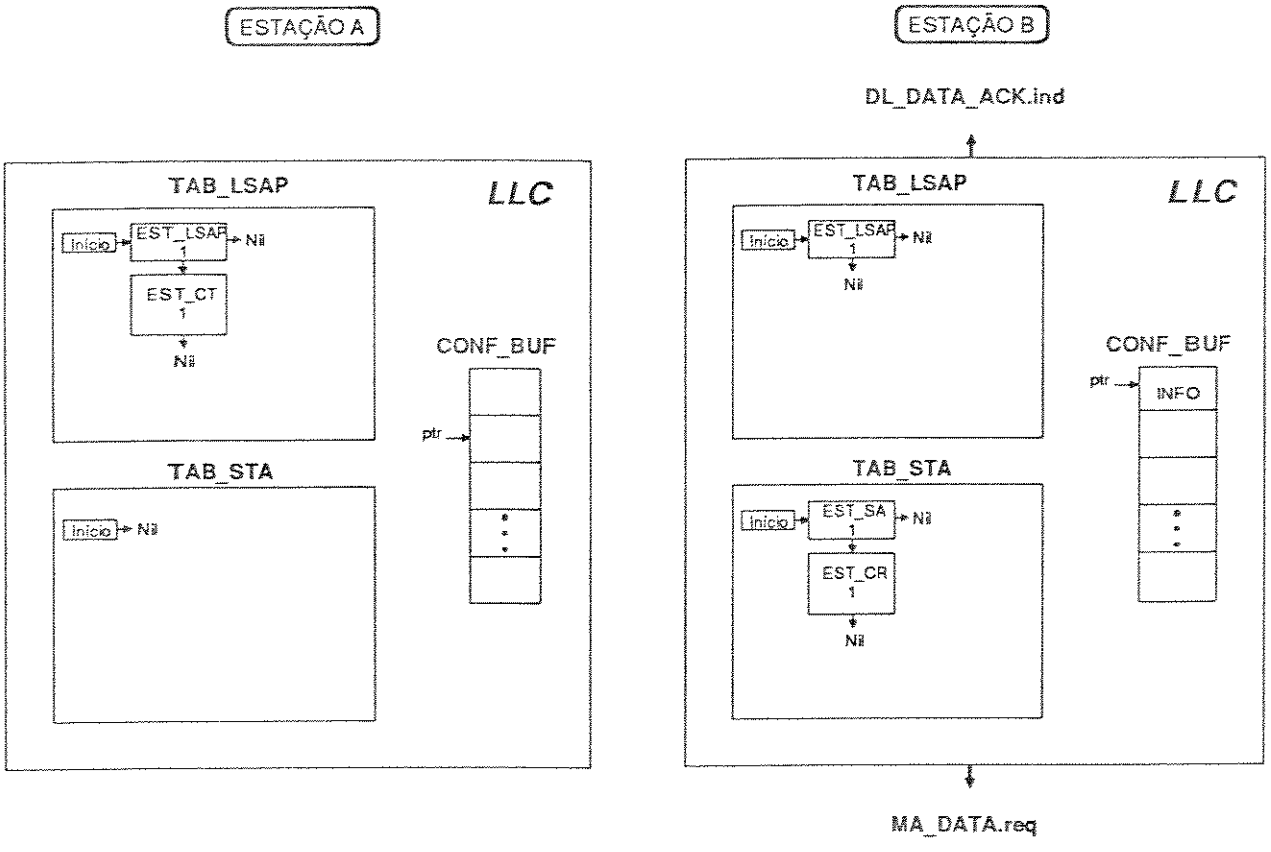

Figura 4.31: LLC_B Envia Dados ao Usuário e Resposta

A estação A recebe o "frame" contendo a resposta do LLC da estação B, que é passada ao LLC pelo MAC através da primitiva *MA_DATA.indication*, como mostra a figura 4.32. Ao mesmo tempo, o LLC da estação B recebe uma primitiva *MA_DATA.confirmation* vinda do MAC local.

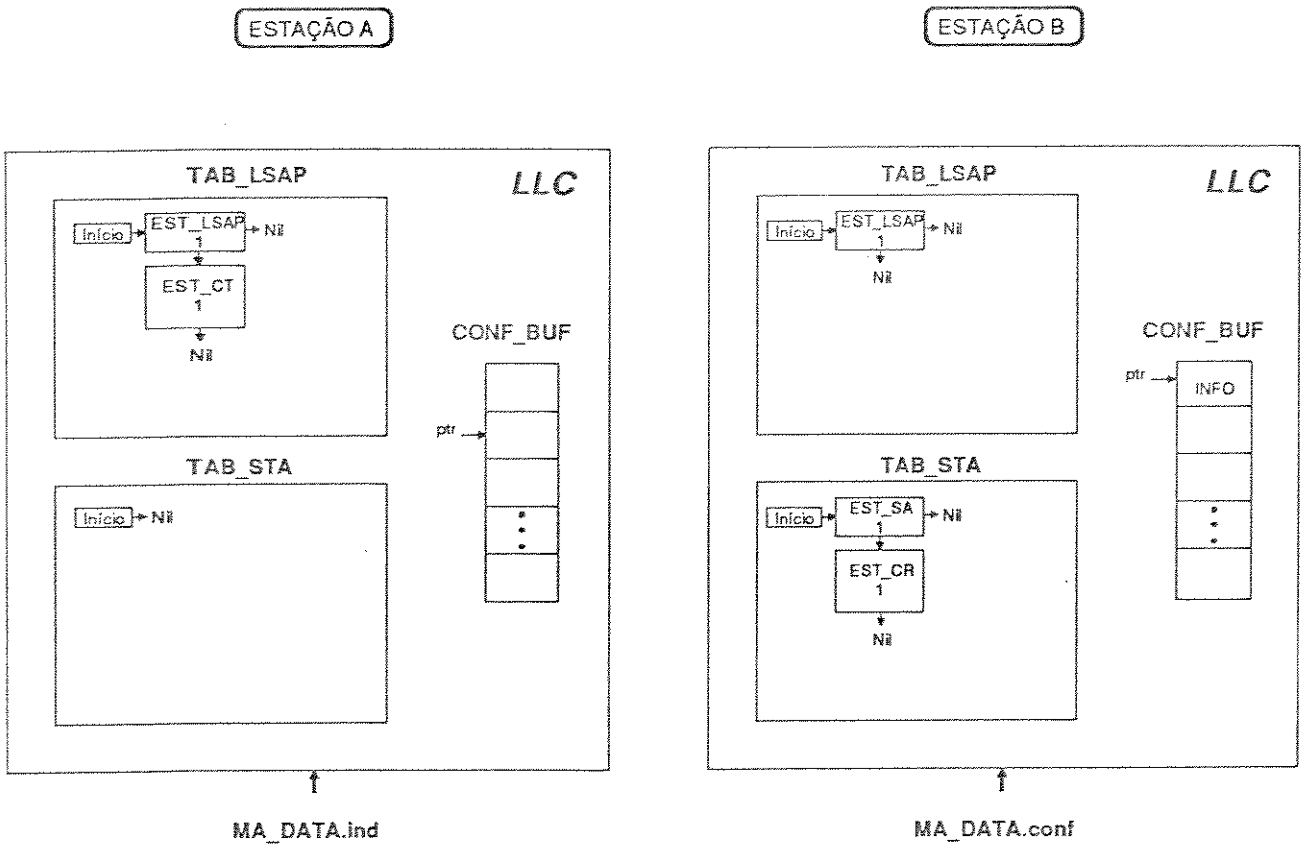

Figura 4.32: Resposta Chega ao LLC_A

Abaixo são descritas as ações executadas nessas estações, em função dos eventos de interface (ver figura 4.33):

- *na estação B*: o LLC obtém em CONF_BUF as informações da transação desenvolvida, mas por se tratar de uma resposta o Componente de Recepção não é endereçado, uma vez que a sua máquina de estado não aguarda uma confirmação da transmissão da resposta (ver tabela TTE_CR em 4.2.3);
- *na estação A*: o LLC obtém os parâmetros DSAP, SA e PRIORIDADE da primitiva recebida e procura em TAB_LSAP o Componente de Transmissão caracterizado por esses valores. Como esse componente já foi criado anteriormente, ele é ativado novamente para que sua máquina de estado processe o novo evento. Desse processamento resulta a primitiva de confirmação *DL_DATA_ACK_STATUS.indication* para a camada de Aplicação, encerrando o ciclo de envio de dados solicitado por ela.

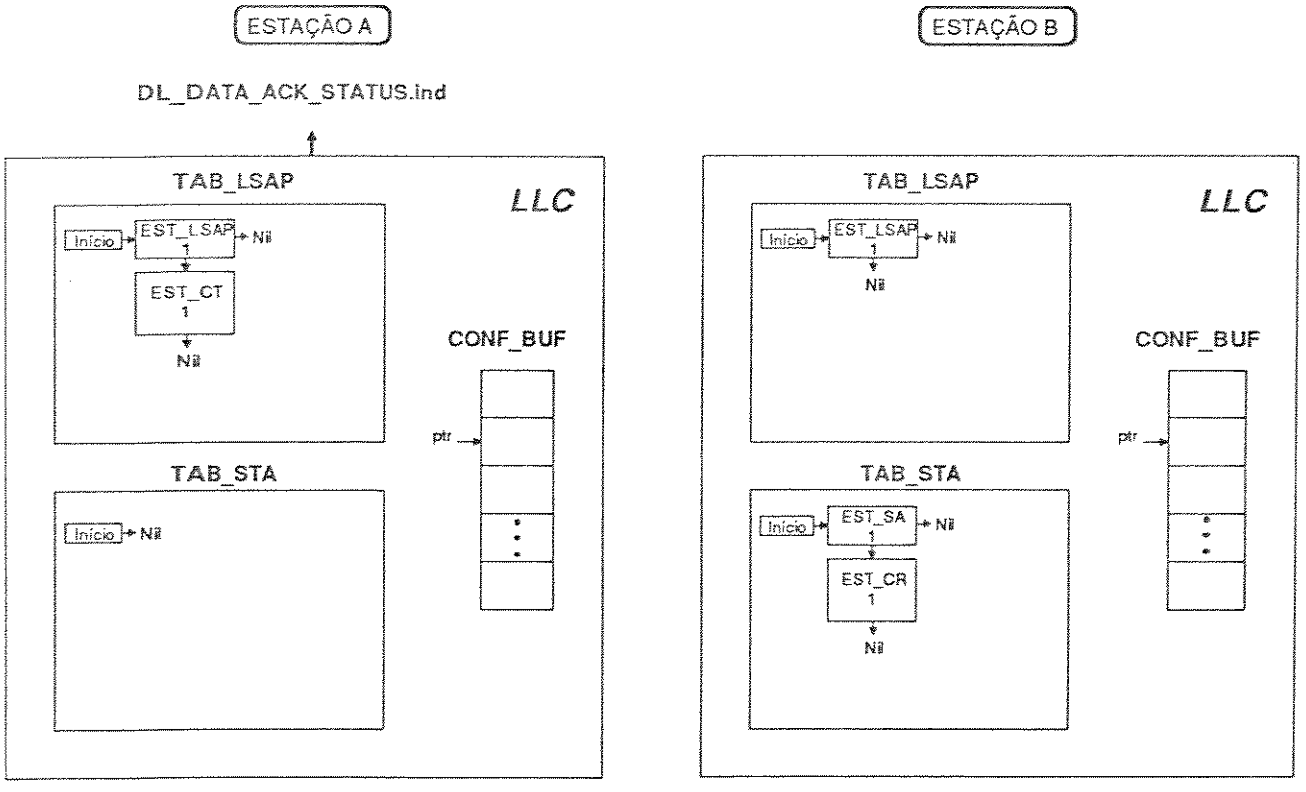

Figura 4.33: LLC_A Envia Confirmação ao seu Usuário

4.3 Avaliação do Desempenho da IRL

Este item apresenta alguns dados referentes à implementação hardware/software da IRL, obtidos com o intuito de avaliar alguns conceitos aplicados e verificar a correção de suas aplicações, sugerindo modificações ao projeto onde elas forem procedentes.

4.3.1 Tamanho dos Programas

Os programas que operam na IRL podem ser subdivididos em 2 grupos: software básico e software aplicativo.

O software básico contém todos os programas que constituem o ambiente multi-tarefas utilizado pelos programas encontrados no software aplicativo, os quais implementam os protocolos executados pela IRL.

O código executável obtido para cada um dos grupos de programas é o seguinte:

Software básico: 10 Kbytes

Software aplicativo: 20 Kbytes

4.3.2 Tempos Obtidos

Foram realizadas medidas de tempo envolvendo duas classes de primitivas: de Gerência e de Aplicação. As primitivas de Gerência pertencem as interfaces Gerência/MAC e Gerência/LLC, enquanto que as de Aplicação pertencem a interface Aplicação/LLC.

Primitivas de Gerência

A execução dessas primitivas é feita localmente pela IRL, ou seja, não resulta na transferência de dados através da rede local. Medindo-se a partir da interrupção gerada pelo hospedeiro na IRL (após a colocação da primitiva no "buffer" de transmissão) até a interrupção gerada pela IRL no hospedeiro (indicando a chegada da primitiva de confirmação no "buffer" de recepção), o tempo de execução dessas primitivas foi:

t_{ger} = 35mseg (máx)

Primitivas de Aplicação

Foi medido o tempo total de execução da primitiva *DL_DATA_ACK request*, tomando-se os mesmos eventos utilizados na medição anterior. A execução dessa primitiva envolve o envio de dados e a recepção de uma resposta por ser um serviço do LLC Tipo 3. Os tempos obtidos na execução dessa primitiva foram:

t_{ti} = 170,5 mseg (máx)

t_{tn} = 150,6 mseg (máx)

O tempo t_{ti} refere-se à primeira execução da primitiva por parte de um Componente de Transmissão. O maior valor de t_{ti} em relação a t_{tn} (tempo total em operação normal) deve-se ao fato de uma instância da tarefa TARCTRANS ser criada dinamicamente para, em seguida, promover o tratamento da primeira primitiva, o que não ocorre nas primitivas subsequentes destinadas ao mesmo Componente de Transmissão.

Para se ter uma noção mais abrangente dos tempos envolvidos nessas transações, serão apresentadas a seguir medições parciais relativas aos tempos de cada uma das fases indicadas na figura 4.34.

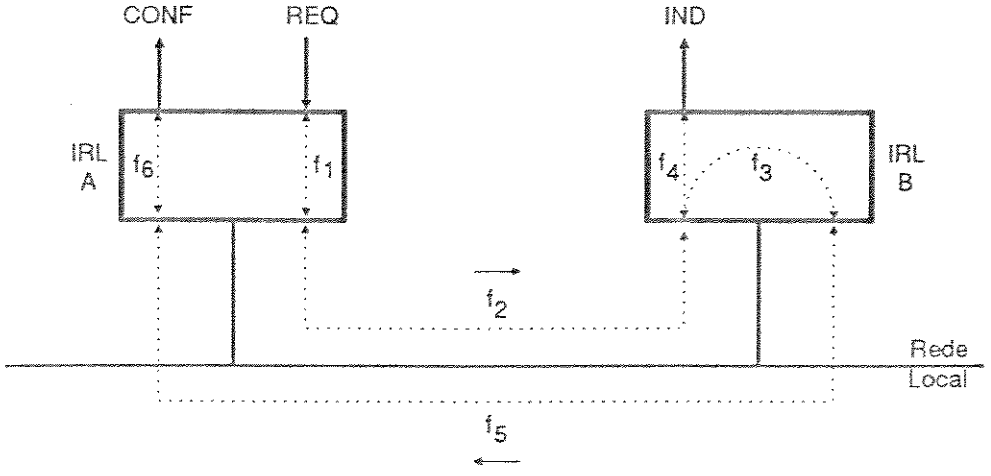

Figura 4.34: Divisão das Fases de uma Transação

A seguir cada fase será descrita e seus tempos máximos medidos serão apresentados. As medições abaixo foram realizadas com 3 nós presentes no anel lógico da rede local, transferindo unidades de dados compostas de 10 bytes.

- *Fase f1*: medida a partir da interrupção do hospedeiro na IRL (após a colocação da primitiva no buffer de transmissão) até a montagem do "frame" resultante, no buffer de transmissão do controlador de rede.

  t_{1i} = 78,5 mseg (Compon. de Transmissão criado)

  t_{1n} = 65,8 mseg

- *Fase f2*: medida desde o final da montagem do "frame" no buffer de transmissão do controlador de rede até a interrupção por ele gerada, indicando o final da transmissão para o outro nó da rede.

  t_2 = 1,1 mseg

- *Fase f3*: medida a partir da interrupção do controlador de rede (indicando a recepção do "frame"), até a montagem do "frame" de resposta no buffer de transmissão do mesmo.

  t_{3i} = 48,2 mseg (Compon. de Recepção criado)

  t_{3n} = 36,5 mseg

- *Fase f4*: medida a partir da interrupção do controlador de rede (indicando a recepção do "frame"), até a geração da interrupção da IRL no hospedeiro (após a colocação da primitiva de indicação no buffer de recepção).

  t_{4i} = 82,1 mseg (Compon. de Recepção criado)

  t_{4n} = 72,4 mseg

- *Fase f5*: medida desde o final da montagem do "frame" de resposta do buffer de transmissão do controlador de rede, até a interrupção por ele gerada, indicando o final da transmissão para o outro nó da rede.

  t_5 = 1,1 mseg

- *Fase f6*: medida a partir da interrupção do controlador de rede (indicando a recepção do "frame" de resposta), até a geração da interrupção da IRL no hospedeiro (após a colocação da primitiva de confirmação no buffer de recepção).

  t_6 = 44,5 mseg

4.3.3 Discussões sobre os Resultados

Com o objetivo de saber-se qual o "overhead" computacional imposto pela utilização do ambiente multi-tarefas, mediu-se o tempo de execução de uma primitiva da interface Gerência/MAC, retirando-se as chamadas ao NMT. Com isso, obteve-se um programa sequencial que recebe a primitiva colocada pelo hospedeiro no "buffer" de transmissão da IRL, executa-a e retorna uma primitiva de confirmação para o hospedeiro. Do programa original foram retiradas as seguintes chamadas ao NMT:

- um "signal" com chaveamento de tarefas;
- um "psend" com chaveamento de tarefas;
- um "getmem";
- um "freemem";
- cinco "wait" para exclusão mútua;
- cinco "signal" para exclusão mútua.

O tempo obtido na execução da primitiva *MA_SET_VALUE.request*, utilizando-se o programa sequencial foi de *4,5 mseg.*

Se o "overhead" obtido neste exemplo (35 - 4,5 = 30,5mseg) fosse dividido igualmente pelo número de chamadas ao NMT feitas no programa original, ter-se-ia uma média de aproximadamente 2 mseg para a execução de cada chamada. Este valor é demasiadamente elevado para um núcleo multi-tarefas que se propõe à aplicações de tempo-real, como é o desejado.

De fato, o NMT utilizado trata-se do núcleo de um sistema operacional de uso geral, não otimizado para aplicações de tempo-real e para redes de comunicação. Além disso, o fato de ele ser quase totalmente implementado na linguagem C e o compilador utilizado não gerar um código objeto otimizado, contribuiu significativamente para a existência desse "overhead".

O NMT foi empregado neste projeto por ser o único núcleo, disponível na época da implementação do software, que possuía as chamadas ao sistema feitas na linguagem C. Certamente, o ideal para se utilizar nesta aplicação, principalmente pelo fato de se trabalhar com um microprocessador de 8 bits, seria um núcleo implementado em linguagem "assembly" com sua interface com o usuário implementada em C.

No que se refere ao sofware aplicativo, algumas alterações podem ser feitas de forma a melhorar o desempenho. A utilização de oito sub-áreas, tanto para a transmissão como para a recepção de primitivas na memória "dual-port" Z80/HOST, foi adotada uma vez que essa área de memória era suficente para armazenar várias primitivas durante todo o processo do tratamento delas. Entretanto, o procedimento de procura de uma primitiva nas sub-áreas e seu tratamento envolve acessos com exclusão mútua nessas sub-áreas, resultando em um "overhead" que agora se verificou ser bastante grande.

Uma solução mais eficiente para esse problema seria o de implementar um "buffer" único de transmissão e outro para a recepção de primitivas, diminuindo-se a área da memória "dual-port" Z80/HOST e aumentando-se a área da memória RAM de trabalho, a qual seria usada pelo NMT para estender o seu "buffer pool". Neste caso, à cada ativação da tarefa TARIRLREQ, por exemplo, uma primitiva seria retirada do TXBUF/HOST e colocada em uma área solicitada ao NMT, para ser encaminhada posteriormente à tarefa que iria tratá-la. Somente com esta alteração, oito semáforos de exclusão mútua dos sub-áreas do TXBUF/HOST seriam eliminados, o mesmo acontecendo com os oito semáforos do RXBUF/HOST.

Ainda que a existência de um único "buffer" comprometa o aspecto de concorrência pelo fato de colocar primitivas com parâmetros de prioridade distintos em uma mesma fila, pela sequência de chegada, ela contribui para uma melhor estruturação dos programas da IRL, uma vez que a manutenção das primitivas no TXBUF/HOST durante seu tratamento resultou em um software bastante dedicado ao hardware, ou seja, sua portabilidade para outros ambientes é muito trabalhosa. É de se esperar que essa alteração torne o software mais simples e, portanto, mais eficiente.

4.4 Ambiente de Desenvolvimento da IRL

Nesta seção serão apresentados os equipamentos e os procedimentos utilizados nas fases de implementação e testes do hardware e software da IRL.

Hardware

As placas dos protótipos da IRL foram confeccionadas em "wire-wrap" a partir de tabelas de conexões geradas pelo sistema CAD da OrCAD Systems Co., empregado na elaboração dos esquemas elétricos.

Os testes de hardware foram feitos, individualmente, em cada um dos módulos apresentados na seção 4.1.2. As rotinas de testes eram geradas na estação de desenvolvimento Tektronix 8561 em linguagem "assembly" do Z80 e enviadas ao emulador Tektronix 8540 (configurado para emulação do Z80) para execução. As rotinas executadas pelo emulador serviam para excitar os circuitos da IRL e através de um analisador lógico Tektronix DAS-9100 era verificada a correção de suas operações e feitas medidas de tempo, onde necessárias.

Software

Na fase de desenvolvimento do software utilizou-se os recursos gráficos de algumas metodologias que auxiliaram na organização do projeto. Os diagramas de dados e controle do GMB ("Graphical Model of Behavior") [MART86] foram usados para especificar as tarefas e os armazenadores de dados do sistema, assim como o fluxo de dados e controle entre esses elementos. O DUD (Diagrama de Uso de Dados) foi usado para descrever os armazenadores de dados, mostrando suas estruturas e elementos. O Pseudo-Código foi utilizado para descrever o processamento realizado pelas tarefas sobre os dados que elas têm acesso.

Como a implementação do software seria feita utilizando-se a linguagem de programação C, foi necessário realizar uma seleção dentre os compiladores C disponíveis para microprocessadores de 8 bits. Existiam três compiladores dessa classe: o da firma BD-Software, o da Manx Software Systems Inc. e o da The Software Toolworks. A escolha recaiu sobre o compilador Aztec C II da Manx Software Systems Inc. por ele permitir a utilização de código reentrante (fundamental para a operação do ambiente multi-tarefas) e por se aproximar mais da implementação da linguagem C definida por Kernigham e Ritchie [KERN86]. Foi utilizada a versão 1.06 do compilador Aztec C II para o sistema operacional CP/M. Este sistema operacional foi emulado em um microcomputador PC/XT utilizando-se o programa CPMV da firma Z-World. Para se obter um código objeto passível de ser gravado em memória EPROM, o compilador utilizado fornece uma biblioteca própria para esse fim.

Para a fase de testes do software da IRL foi implementado um programa para simular o envio e a recepção de primitivas entre o hospedeiro e a IRL. Interagindo com esse programa o usuário do hospedeiro podia gerar primitivas para as interfaces Gerência/MAC, Gerência/LLC e Aplicação/LLC e observar a chegada de primitivas de confirmação referentes aos serviços requisitados ou de primitivas de indicação resultantes da recepção de mensagens de outras estações.

A depuração do software da IRL foi trabalhosa pois teve que ser feita analisando-se a execução do código objeto em linguagem "assembly", com o auxílio do "disassembler" do emulador ou do analisador lógico.

O software não pode ser depurado no hospedeiro, a nível do código fonte, por vários motivos. No hospedeiro de 8 bits não se acha disponível um depurador para a linguagem C. No hospedeiro de 16 bits, como o PC, eles existem (por exemplo o CodeView da Microsoft), porém exigiria muito trabalho adaptar o NMT para operar no PC e simular o hardware da IRL (o controlador de rede, por exemplo, possui uma operação bastante complexa).

Capítulo 5

Conclusões

As vantagens advindas com a utilização de sistemas distribuídos em aplicações industriais, no que se refere à flexibilidade, desempenho, confiabilidade, custo e disponibilidade, são amplamente conhecidas.

A aplicação de técnicas de manufatura integrada como "just-in-time", MRP II e outras, em sistemas distribuídos, incentivaram empresas à adotarem o CIM ("Computer Integrated Manufacturing") com o objetivo de obter maior competitividade no mercado, oferecendo produtos a custos mais baixos, melhor qualidade e entregas mais rápidas.

Entretanto, existe um elemento que viabiliza a existência dos sistemas distribuídos: a rede local de comunicação de dados. Através dela ou de uma hierarquia de redes locais interligadas, uma empresa consegue integrar os diversos componentes do seu ciclo industrial. Por outro lado, a maioria dessas empresas deparam com o problema de interligação de equipamentos de diferentes fornecedores, com recursos de comunicação incompatíveis entre sí. Este fato tem aumentado o interesse dessas empresas na utilização de normas internacionais para as redes de comunicação, uma vez que, como usuárias, elas são as que mais têm a ganhar com a imposição dessas normas aos fabricantes de equipamentos.

Foi com essa linha de pensamento que surgiram as especificações MAP e TOP, lideradas pelas empresas GM e Boeing e, atualmente, encontra-se em desenvolvimento a especificação para o "Field-Bus".

O projeto descrito neste trabalho resultou na implementação de uma interface de rede local com características apropriadas para aplicações industriais. Dessas características se destacam:

- topologia do tipo "barramento";
- protocolo de acesso ao meio físico "token-passing";
- alta taxa de transmissão (1 Mbit/seg);
- arquitetura de camadas reduzida (similar ao Mini-MAP);
- utilização de um ambiente multi-tarefas para contemplar a prioridade de mensagens.

A utilização, na subcamada MAC, de um "chip" controlador de rede que não implementa o padrão IEEE 802.4, fez com que este projeto gerasse, por assim dizer, mais uma "rede proprietária". Porém, isso aconteceu em decorrência da época em que o projeto teve início e da falta de recursos humanos e materiais encontrada no Instituto de Automação do CTI, que impossibilitou a atualização do hardware da IRL. Deve-se salientar, por outro lado, a preocupação em se fazer com que o restante do projeto fosse implementado seguindo-se as normas internacionais existentes.

Além da confecção de um hardware específico para comunicação em redes locais, conseguiu-se com este projeto testar uma proposta de arquitetura de software para aplicações industriais. Empregou-se, para isso, o modelo do servidor de multiprocessos ("Multiprocess Server") [SVOB85], onde uma tarefa é responsável por uma conexão, que neste caso pode ser mapeada em um Componente de Transmissão ou Recepção. Apesar deste modelo ser usado geralmente a nível de especificação, a sua utilização foi bastante positiva, não só pela melhor modularidade proporcionada ao software do sistema, como também pela introdução de concorrência no tratamento de mensagens de prioridades distintas que transitam por uma rede industrial.

Como pontos negativos, verificou-se a inadequabilidade de se usar um ambiente multi-tarefas de propósitos gerais, na implementação de protocolos de comunicação. Alguns serviços desejados para esta aplicação, tais como gerenciamento eficiente de memória, tratamento de eventos e temporização na sincronização e comunicação entre tarefas, não eram disponíveis no NMT, obrigando a um aumento na complexidade dos programas.

A subestimação do tamanho dos programas dos protocolos de comunicação e do "overhead" resultante da utilização do NMT, no momento de definição das capacidades de processamento e memória do hardware da IRL, foi outro ponto negativo. Isso gerou um software pouco portável para outros equipamentos e com desempenho bastante baixo, para as aplicações a que se destina. Por outro lado, várias porções dos programas gerados podem ser recodificadas de forma a melhorar o desempenho, além das alterações anteriormente citadas no item 4.3.3.

Como visto em 4.4, utilizou-se neste projeto alguns recursos de metodologias para o desenvolvimento do software da IRL. Porém, o uso superficial dessas metodologias, executado de forma manual, resultou na falta de um elemento do qual a maioria dos projetos de software carece: uma documentação de todas as fases do desenvolvimento que um projeto deveria ter à rigor. Mesmo sendo este um projeto de pequeno porte, é fortemente aconselhável o uso de ferramentas integradas, tipo CASE, na elaboração do desenvolvimento de software. A utilização do sistema EPOS, por exemplo, tem se mostrado eficiente na fase de especificação da implementação de protocolos, nos trabalhos sendo realizados no Departamento de Computação e Automação da Faculdade de Engenharia Elétrica da UNICAMP [INAZ90].

O aperfeiçoamento deste projeto, atualmente, requeriria a confecção de um novo hardware, através do qual se adotaria completamente as normas internacionais para as camadas de protocolo. Esse processo se iniciaria com a adoção de um "chip" controlador de rede padrão IEEE 802.4 (por exemplo o Motorola 68824). Com isso, o software da subcamada

LLC seria bastante simplificado, uma vez que seriam utilizadas as tabelas de transição de estados da operação Tipo 3 do LLC contidas no Apêndice A do documento [ISO86] (eliminando-se, por exemplo, os temporizadores de confirmação dos Componentes de Transmissão).

O uso de microprocessadores de 16/32 bits é obrigatório para se atingir o desempenho desejado para aplicações de tempo-real com a utilização de um ambiente multi-tarefas. O aumento da potência computacional resultante do uso desses microprocessadores, aliado com suas grandes capacidades de endereçamento de memória, permitirá agrupar todas as três camadas que compõem a arquitetura Mini-MAP na mesma placa de rede local, liberando o hospedeiro para executar apenas os seus programas aplicativos, com um mínimo de "overhead" destinado à comunicação de dados.

Bibliografia

[APPE87] L. Appezato; *Um Processador de Comunicação de Rede Local para Controle de Processos*; V Simpósio Brasileiro de Redes de Computadores, São Paulo; Abril 1987.

[APPE89] L. Appezato, et al; *Implementação de um Sistema de Comunicação Industrial no CTI*; Seminário Franco-Brasileiro em Sistemas Informáticos Distribuídos, Florianópolis; Setembro 1989.

[BABB87] M. P. Babb; *Networking Requires a Range of Solutions*; Control Engineering, 34(11):13; October 1987.

[BENN82] S. Bennett, D. A. Linkens; *Computer Control of Industrial Processes*; IEEE/Peter Peregrinus; 1982.

[COME84] D. Comer; *Operating Systems Design, the XINU Approach*; Prentice-Hall Software Series; 1984.

[CROW85] R. Crowder; *The MAP Specification*; Control Engineering, 32(11):22-25; October 1985.

[CTIA84] Centro Tecnológico para Informática - IA; *Documento Interno: Projeto Estruturado de Programas*; 1984.

[CUSH84] R. H. Cushman; *Hands-on Network-Design Project Gives Insight into LAN Features*; EDN,29(6):219-230; March 1984.

[DETI88] United Kingdom. Department of Trade and Industry; *The Computer Integrated Organization: Some Business and Technical Issues for the OSI-MAP/TOP Solution*; Society of Manufacturing Engineers (SME); 1988.

[GEDP84] Government EDP Standards Commitee; *A Guide to Open Systems Interconnection*; Computer & Standards, 3(1984):171-193; North-Holland; January 1984.

[GEMO88] General Motors Co; *Manufacturing Automation Protocol (MAP) Specification - Version 3.0*; August 1988.

[GIOZ86] W. F. Giozza, et al; *Redes Locais de Computadores: Tecnologia e Aplicações*; São Paulo:McGraw-Hill; 1986.

[HARR84] Harris Semiconductor Co.; *HD 6409 Data Sheet*; September 1984.

[HOLL86] J. Hollingum; *The MAP Report: Manufacturing Automation Protocol*; IFS/Springer-Verlag; 1986.

[IEC86] IEC; *Field Bus Standard for Use in Industrial Control Systems: Functional Requirements - Draft for National Comment*; July 1986.

[IEEE84] IEEE; *IEEE Standard 802.2: Logical Link Control*; The IEEE Inc; 1984.

[IEEE85] IEEE; *IEEE Standard 802.4: Token-Passing Bus Access Method and Physical Layer Specifications*; The IEEE Inc; 1985.

[INAZ90] P. C. M. Inazumi; *Implementação do Protocolo da Camada de Apresentação OSI*; Tese de Mestrado em andamento, FEE, UNICAMP; 1990.

[ISA88] ISA; *Standards and Practices for Instrumentation: PROWAY-LAN Industrial Data Highway*; vol. 2, 9th edition; 1988.

[ISO82] ISO; *Information Processing Systems - Open Systems Interconnection - Basic Reference Model*; ISO/DIS 7498; April 1982.

[ISO86] ISO; *ISO DIS 8802/2: Logical Link Control - Acknowledged Connectionless Service - Draft Proposed Addendum*; ANSI Inc; 1986.

[JDAY83] J. D. Day, H. Zimmermann; *The OSI Reference Model*; Proceedings of the IEEE, 71(12):1334-1340; December 1983.

[KAMI86] M. A. Kaminski Jr.; *Protocols for Communicating in the Factory*; IEEE Spectrum, p. 56-62; April 1986.

[KERN86] B. W. Kernighan, D. M. Ritchie; *C: A Linguagem de Programação*; Ed. Campus, EDISA; 1986.

[MART86] J. M. de Martino; *Um Ambiente GMB* para o Desenvolvimento de Sistemas Distribuídos de Controle Digital*; Dissertação de Mestrado em Eng. Elétrica, FEE, UNICAMP; Julho 1986.

[MEND88] M. J. Mendes, M. F. Magalhães; Redes Industriais e o Projeto de Padronização MAP/TOP; SBA Controle e Automação, 2(1):56-70; Março 1988.

[MTUG88] MAP/TOP Users Group; *Technical and Office Protocol (TOP) Specification - Version 3.0*; August 1988.

[NELS83] J. Nelson; *802: A Progress Report*; Datamation, 29(9):136-152; September 1983.

[STAL84] W. Stallings; *Local Networks: An Introduction*; Macmillan; 1984.

[SVOB89] L. Svobodova; Implementing OSI Systems; IEEE Journal on Selected Areas in Communications, 7(7):1115-1130; September 1989.

[TANE89] A. S. Tanenbaum; *Computer Networks*; Prentice-Hall; 1989.

[ULME88] M. Ulmer; *Field-Bus, Um Novo Padrão de Rede Local Industrial ?*; Automação e Indústria, 2(6):16-18; Maio 1988.

[WEST84] Western Digital Co; *Communication Products Handbook*; June 1984.

[ZILO82] Zilog Inc.; *Zilog Data Book*; 1982.