UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA DE CAMPINAS

DEPARTAMENTO DE ENGENHARIA ELETRICA

CAMPINAS, ABRIL DE 1985

Este exemplar corresponde à redação final da tese defendida por José Marcos Silva Nogueira e aprovada pela comissão julgadora em [illegible]

ASPECTOS DE COMUNICAÇÃO EM SISTEMAS DISTRIBUIDOS PARA APLICACOES DE AUTOMAÇÃO INDUSTRIAL

POR JOSE´ MARCOS SILVA NOGUEIRA

Orientador de Tese: Prof. Dr. Manuel de Jesus Mendes

13/85

Tese de Doutorado em Engenharia Elétrica apresentada à Faculdade de Engenharia de Campinas.

A

Ângela.

A

Maíra.

Agradecimentos

Agradeço a todos aqueles que de uma forma ou de outra contribuíram para que este trabalho chegasse a termo, em particular ao Prof. Manuel de Jesus Mendes pela orientação e ao Prof. Newton Alberto de Castilho Lages que muito contribuiu com sugestões e trocas de idéias.

Agradeço também às entidades Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelas bolsas de estudo que tornaram possível o meu curso de doutorado.

A minha esposa, Ângela, sou muito grato pela sua compreensão e paciência, virtudes fundamentais sem as quais este trabalho seria muito árduo. Ademais, agradeço a sua solidariedade, pela qual interrompeu temporariamente sua carreira profissional.

Folha de Aprovação

Aspectos de Comunicação em Sistemas Distribuídos
para Aplicações de Automação Industrial

Tese de Doutorado em Engenharia Elétrica apresentada por
José Marcos Silva Noqueira
e aprovada pela banca examinadora constituida dos Senhores

Prof. Dr. Newton Alberto de Castilho Laqes

Prof. Dr. João Eduardo de Resende Dantas

Prof. Dr. Márcio Luiz de Andrade Netto

Prof. Dr. Clésio Luiz Tozzi

Prof. Dr. Manuel de Jesus Mendes
- Orientador -

Departamento de Engenharia Elétrica
Faculdade de Engenharia de Campinas
Universidade Estadual de Campinas
Campinas, 30 de abril de 1985

R E S U M O

A área de automação industrial vem experimentando um notável progresso nos últimos anos. Entre os fatores que para isso concorrem está o desenvolvimento da tecnologia da computação. Verifica-se nos dias de hoje uma tendência de integração dos vários subsistemas e atividades que constituem um sistema complexo de automação industrial. Neste sentido, adquire singular importância a interligação e comunicação entre elementos de computação. Este trabalho de tese se insere neste contexto, tratando dos aspectos de comunicação em sistemas distribuídos para aplicações em automação industrial, as quais se dividem em dois ramos principais, o controle de processos e a automação da manufatura.

Inicialmente são apresentados os conceitos envolvidos em sistemas distribuídos e em redes locais de comunicação. Em seguida procura-se identificar e caracterizar os componentes dos sistemas distribuídos para aplicações em controle de processos industriais e para aplicações na automação integrada da manufatura.

A problemática da integração das atividades da automação industrial e, por consequência, de seus subsistemas, é muito vasta. De interesse no caso são os problemas relacionados com a comunicação entre os componentes que viabilizam esta integração. Mesmo certas atividades particulares, como, por exemplo, o controle distribuído de processos e a manufatura através de sistemas flexíveis, demandam um tratamento sob o enfoque de sistemas distribuídos. No nível mais baixo da comunicação, onde se inserem as redes locais, existem problemas de topologia e de controle de acesso ao meio, bem como de protocolos de comunicação e sua padronização. Num nível intermediário, de suporte à comunicação das aplicações, existe o problema de se prover mecanismos para comunicação e sincronização entre processos computacionais. No nível superior, diretamente relacionado com as aplicações, quase tudo está por ser feito. Ajunta-se a isto o

fato que as demandas de comunicação das aplicações têm perfis heterogêneos em termos de quantidade de dados e requisitos de tempo. Alguns desses problemas são tratados neste trabalho através de um exemplo ilustrativo de aplicações de controle distribuído de processos. São levantadas as caracteristicas e necessidades de comunicacação em termos de demanda de tráfego e tipo de informação transmitida. As aplicações são caracterizadas do ponto de vista dos programas constituintes e um modelo de programas paralelos é elaborado. Especifica-se, até o nível de implementação, um subsistema para dar suporte e prover mecanismos de comunicação e sincronização entre processos comunicantes. Este subsistema corresponde a uma camada de serviços de nível intermediário na arquitetura de sistemas distribuídos. Finalmente é apresentado um ferramental para auxílio na análise de desempenho de sistemas distribuidos, usando técnicas de simulação e de teoria de filas. Os aspectos de desempenho são importantes no desenvolvimento de sistemas distribuídos em geral, e, em particular, no desenvolvimento de camadas individuais de serviço.

S U M A R I O

1. APRESENTAÇÃO

A automação industrial vem experimentando nos últimos anos um grande progresso. Com o desenvolvimento das tecnologias de microeletrônica, microprocessadores e comunicação, bem como da engenharia de software, tem sido possível, cada vez mais, tanto a automatização com computadores das diversas funções envolvidas na produção industrial, como a integração dessas mesmas funções. Essas funções, de naturezas variadas, vão desde o projeto de produtos e sistemas, passando pelo planejamento e administração, até o controle de processos industriais.

Por outro lado, progresso semelhante ocorre na área de interligação de computadores e comunicação de dados. As redes de computadores são hoje uma realidade, embora muito haja o que ser feito - em particular, nas redes locais de comunicação para interligação de equipamentos e/ou computadores. Há aí uma demanda para a padronização da comunicação e faltam mecanismos de comunicação para suportar aplicações distribuídas. Estas são áreas com grande atividade de pesquisa e desenvolvimento.

A integração das funções de automação industrial está em parte condicionada à existência de uma ou mais redes locais de comunicação. Mesmo certas funções, como o controle distribuído de processos complexos, por si só justificam uma rede local para sua implementação.

O presente trabalho de tese se insere neste contexto, tratando de aspectos de comunicação em sistemas distribuídos para automação industrial em geral e, em particular, de comunicação para implementação de sistemas de controle distribuído de processos industriais.

O segundo capítulo contém uma apresentação dos conceitos envolvidos em sistemas distribuídos e redes de computadores. Depois apresentam-se as redes locais de comunicação, incluindo aspectos do meio de comunicação e de padronização. Em seguida é

feita uma caracterização dos sistemas para controle distribuído de processos industriais, bem como é concebido um modelo de referência baseado em redes locais de comunicação. Ao final há o tratamento sobre as funções da automação da manufatura e sua integração através de redes locais de comunicação.

No terceiro capítulo trata-se dos aspectos e problemas de comunicação em aplicações distribuídas de automação industrial. Toma-se, a título de exemplo ilustrativo, as aplicações de controle distribuído de processos. Inicialmente são levantadas as características e necessidades de comunicação em termos de demanda de tráfego e do tipo de informação trocada entre os componentes da aplicação. Em seguida caracteriza-se a aplicação do ponto de vista dos programas aplicativos constituintes e elabora-se um modelo de programas paralelos. Este busca dar uma visão uniformizada da aplicação e da comunicação entre as partes, incluindo aí aspectos de sincronização da comunicação. Para dar suporte à comunicação da aplicação, especifica-se um subsistema de comunicação que implementa os requisitos da aplicação então levantados. Do ponto de vista da arquitetura de sistemas distribuídos, este subsistema corresponde a uma camada de serviços. Finalmente a especificação anteriormente feita é apresentada com um pouco mais de detalhe, visando implementações reais.

O quarto capítulo visa a análise de desempenho de sistemas distribuídos que, embora tenha sido orientada para as aplicações particulares do capítulo 3, pode ser extendida para outras aplicações de automação industrial. A análise foi feita por simulação, para o que desenvolveu-se um simulador, e por ferramentas matemáticas de verificação analítica. Como não é o objetivo central do trabalho, o assunto é apresentado de forma sucinta.

2 SISTEMAS DISTRIBUIDOS PARA AUTOMAÇAO INDUSTRIAL

2.1 INTRODUÇAO

A fim de se chegar `a apresentação das caracteristicas e dos problemas envolvidos em sistemas distribuidos para aplicação na automação industrial, usa-se a estratégia de abordagem do assunto a partir de aspectos gerais no sentido dos especificos. Assim, procura-se caracterizar os sistemas distribuidos de uma maneira genérica e apresentar sua arquitetura. Em seguida, consideram-se as redes locais para comunicação de computadores, cada vez mais usadas nos sistemas distribuidos tipicos usados em aplicacões de controle de processos e automação da manufatura. No final, trata-se especificamente de sistemas de controle tempo real.

2.2 SISTEMAS DISTRIBUIDOS

Para apresentar os conceitos gerais envolvidos em sistemas distribuidos, esta seção trata primeiramente das motivacões que têm levado à rápida evolução destes sistemas. Em seguida, há uma discussão sobre o que é esperado destes sistemas, ou seja, quais os objetivos a alcançar. Finalmente, na falta de uma definição consensual do tema, levantam-se suas caracteristicas mais importantes.

2.2.1 Avanços Tecnológicos e Necessidades dos Usuários

O atual estágio de desenvolvimento e utilização de sistemas de processamento distribuido deve-se principalmente a dois fatores que se complementam: o primeiro é o avanço tecnológico experimentado, nestas últimas décadas, na microeletrônica e nas comunicações; o segundo fator é decorrente das necessidades computacionais dos usuários, cada vez mais sofisticadas. Estas

caminham no sentido de exigir, em lugar dos sistemas de computação baseados em computadores centrais, sistemas baseados em vários computadores separados mas interconectados entre si com o objetivo de fazer trabalhos comuns. Este "casamento" da tecnologia com as necessidades tem estimulado e viabilizado o surgimento de tais sistemas.

Do avanço tecnológico da microeletrônica, pode-se dizer que tem sido extremamente rápido e redutor de custos. A redução de custos de componentes lógicos tem sido, nos últimos anos, na base de 1/10 a cada 3 anos. A tendência é o quadro se manter ou melhorar ainda mais. A capacidade das pastilhas tem aumentado de forma semelhante. Basta dizer, por exemplo, que por volta de 1973 se conseguia armazenar 4k bits em uma pastilha de memória e atualmente já há pastilhas de leitura/escrita com 256k bits de capacidade e pastilhas de leitura apenas com capacidade de 1 Mbits. Desenvolvimento semelhante tem havido com outros componentes lógicos, em particular os processadores, em termos de potencial de cáculo e de velocidade de processamento.

Para que elementos de computação possam cooperar trocando informação, é necessário que estejam interconectados por um meio físico de comunicação. A tendência de se ligar computadores não é nova. Novo é o enfoque generalizado dado aos sistemas de interconexão, bem como o seu porte físico. Distinguem-se dois ramos de tecnologia de comunicação que contemplam os sistemas distribuídos. Primeiro tem-se os sistemas baseados em redes telefônicas (ou do mesmo tipo) de média e longa distância. Estes possibilitam a interconexão a nível de cidade, país ou mesmo a nível internacional. São recursos em geral compartilhados por uma diversa gama de usuários. Existem iniciativas de organizações internacionais de padronização (ISO, CCITT, etc), com o objetivo de padronizar o uso e o acesso a estes recursos. Alguns resultados concretos já foram apresentados, aceitos e implementados. O segundo ramo da tecnologia de comunicação contempla a interconexão de computadores no âmbito de distâncias curtas, onde os elementos de computação interconectados se distribuem geograficamente ao longo de um edifício, um campus,

uma fábrica ou mesmo uma cidade. São os sistemas de interconexão usados nas redes locais de comunicação, sistemas de origem relativamente recente, com um único usuário. São usadas, para a interconexão, diversas tecnologias, desde as simples e dominadas (par trançado de fios, cabo coaxial, rádio transmissão) até as novas e complexas (fibras óticas).

Quanto às necessidades computacionais de usuários, observa-se uma tendência de demanda crescente de sistemas com processamento distribuído, tanto para novas aplicações como para substituição de aplicações existentes, baseadas em sistemas centralizados. Alguns aspectos concorrem para que isto aconteça. O conceito de "centro de computação", predominante ainda hoje, onde um único computador faz todo o serviço, traz consigo a prática de que os usuários é que devem ir ao computador, ao invés de o computador ir ao usuário. De outra forma, nos sistemas centralizados o processamento não é feito onde os dados estão, e sim onde o computador está. Existe uma variada gama de aplicações, de naturezas comercial, industrial e científica, nas quais grande parte do processamento envolvido pode ser mais eficientemente executado nos locais onde são gerados os dados.

Muitas organizações apresentam uma estrutura funcional não centralizada. O uso de sistemas centralizados impõe um estilo centralizado de gerenciamento, o que não é desejado. Como consequência, há uma relutância em se adotar tais sistemas e uma procura cada vez maior de sistemas com múltiplos computadores interconectados, que reflitam a estrutura da organização.

Aliada a estes fatores, existe a necessidade permanente de redução de custos. A descentralização do processamento contribui para a diminuição de custos em dois aspectos. Primeiro, reduz o custo de transporte de informação quando o processamento em sua maior parte é feito localmente - dados que transitam são apenas os estritamente necessários para a boa execução do trabalho. Segundo, o uso de um conjunto de micro ou mini computadores interconectados, em substituição a um sistema de grande porte de desempenho equivalente, leva a preços mais baixos. Neste sentido até mesmo o custo de software decresce, pois sistemas de grande

porte exigem "software" de suporte grande, complexo e de custosa manutenção.

Estes fatores, coletivamente, levam à conscientização de que novos sistemas são necessários em substituição aos tradicionais. Ao que tudo indica, o conceito que se firma nesta direção é o de processamento distribuído. Todavia, há uma série de requisitos que são esperados ou desejados dos sistemas distribuídos e também um conjunto de vantagens possíveis de alcançar, como será visto no próximo item.

2.2.2 Objetivos e Vantagens dos Sistemas Distribuídos

Os sistemas distribuídos devem atender a uma série de objetivos que se constituem em vantagens sobre sistemas centralizados. Os mais importantes e de caráter geral são relacionados a seguir.

Disponibilidade

Os avanços tecnológicos e conhecimentos adquiridos na construção de "hardware" e "software" têm possibilitado a obtenção de produtos mais confiáveis. Porém, em qualquer sistema físico, a probabilidade que ocorram falhas, erros ou faltas é sempre maior que zero. Uma maneira de aumentar a confiabilidade é através do uso de fontes alternativas de suprimento ou redundância: quando um componente não pode estar em operação, outro toma seu lugar e passa a executar suas funções. Em sistemas centralizados, a perda de um recurso crítico é séria, podendo trazer consequências desastrosas ou mesmo intoleráveis para certas aplicações. A disponibilidade de tais sistemas é baixa, pois depende diretamente se o recurso está operacional ou não. Sistemas distribuídos podem ser organizados de tal modo que exista inspeção mútua entre os componentes e procedimentos de diagnóstico e recuperação. Isto implica muitas vezes na substituição de um componente por outro para executar suas funções. Neste caso, o sistema como um todo continua em

funcionamento, mesmo que degradado, a despeito de falhas. A disponibilidade pode então ser muito alta, função direta do grau de redundância e da capacidade dos referidos procedimentos.

Compartilhamento de recursos

Este objetivo significa a disponibilidade para qualquer usuário, a despeito de sua localização geográfica, de recursos como programas, dados, elementos de processamento e outros dispositivos físicos. Requer muitas vezes um controle a nível de sistema para alocação ótima e dinâmica de recursos, bem como o compartilhamento ótimo desses recursos.

Extensibilidade

Também chamado de "crescimento incremental", diz respeito à possibilidade de construção de sistemas que podem ser facilmente adaptados a novos ambientes e ter seu porte alterado sem interrupção do seu funcionamento. Mudanças ambientais significam troca de componentes, expansão e redução de configuração, bem como expansão do conjunto de serviços oferecidos. Desse modo, pode partir-se de uma configuração inicial de baixo custo e, com expansões graduais de baixo custo por vez, chegar-se a configurações com mais componentes.

Este objetivo implica na concepção de arquiteturas modulares. Traz consigo algumas vantagens, entre as quais a concepção mais simples (componentes especializados e menos complexos), maior facilidade de instalação (incremental) e maior facilidade de manutenção.

Melhoria de desempenho

A melhoria de desempenho esperada para os sistemas distribuídos baseia-se em dois fatos: primeiro, a relação preço/desempenho dos pequenos computadores é superior à dos grandes. Por exemplo, um grande computador, 10 vezes mais veloz que os melhores microcomputadores, custa centenas ou milhares de vezes mais.

Segundo, o desempenho em sistemas centralizados sofre uma perda significativa devido a problemas de contenção de recursos e de chaveamento de contexto (estes sistemas são multiprogramados). Um sistema distribuido executa suas tarefas paralelamente, com o minimo de contenção e chaveamento. Isto tem consequências agradáveis para os usuários, tais como acesso direto e simplificado ao sistema de computação, redução do tempo de retorno para atividades em "batch" ("turn around time"), e melhoria do tempo de resposta para atividades em tempo real.

2.2.3 Caracterização

Pode-se dizer que não há ainda entre os especialistas da área uma definição de sistemas distribuidos que seja consensual. Existem definições desde as mais frouxas e amplas, que permitem encaixar qualquer sistema que apresente algun tipo de distribuição, até definições as mais estritas, nas quais é difícil encaixar os sistemas existentes. Por isso, optou-se, neste trabalho, por levantar pontos que caracterizem os sistemas distribuidos, sem o compromisso de uma definição rigorosa. As caracteristicas são levantadas a partir de definições e caracterizações existentes na literatura.

Uma definição bastante geral é apresentada por Liebowitz & Carson (1978), elaborada levando-se em consideração várias visões sobre o tema:

> "um sistema distribuido é um sistema de computação em que as funções computacionais são alocadas entre vários elementos físicos de computação. Estes podem estar geograficamente próximos ou separados uns dos outros".

Na mesma linha de generalidade, Tanenbaum (1981) define o que ele chama "rede de computadores":

> "uma rede de computadores significa uma coleção de

computadores autônomos e interconectados. Dois computadores são ditos interconectados quando são capazes de trocar informação, o que pode ocorrer através de um meio físico de comunicação. Este é composto, por exemplo, de fios de cobre, micro-ondas, fibras óticas ou satélites espaciais. O requisito de autonomia exclui sistemas nos quais exista um claro relacionamento do tipo mestre/escravo entre os computadores".

Uma definição rigorosa e estrita, e por isso bastante polêmica, é apresentada por Enslow (1978 b). Todavia ele faz a ressalva que sua definição é para dar subsídios a projetos de novos sistemas, de modo a serem alcançados os objetivos citados anteriormente. A definição tem 5 componentes:

"uma multiplicidade de recursos computacionais de uso geral, incluindo recursos físicos e lógicos, que podem ser dinamicamente encarregados da execução de tarefas específicas. Homogeneidade de recursos físicos não é essencial";

"uma distribuição geográfica desses componentes físicos e lógicos, os quais interagem através de uma sub-rede de comunicação de dados";

"um sistema operacional de alto nível que verifique e integre o controle dos componentes distribuídos. Cada processador individual pode ter seu próprio sistema operacional";

"tansparência de sistema que permita a requisição de serviços apenas por nome, sem identificação de quem será o executor";

"autonomia cooperativa, caracterizando a operação e interação dos recursos físicos e lógicos. Implica na não existência de hierarquia de controle dentro do sistema".

Esta definicão requer um sistema operacional a nivel global, o que não é requisito para as definicões de Liebowitz/Carson e de Tanenbaum. Porém estas duas comportam a definicão de Enslow, ou seja, um sistema distribuido concebido conforme propõe Enslow se encaixa nas duas primeiras definições apresentadas.

No contexto deste trabalho, ao invés de se utilizarem definicões amplas ou estritas como aquelas apresentadas acima, optou-se por considerar como sistema distribuido aquele que apresenta um conjunto de caracteristicas de distribuição em termos fisicos e lógicos. As especificacões destas caracteristicas são, segundo LeLann (1981) :

(1) um sistema distribuido inclui um número arbitrário de processos de sistema e de usuário;

(2) a arquitetura é modular, consistindo de um número possivelmente variável de elementos de processamento;

(3) a comunicacão dá-se por troca de mensagens através de uma estrutura de comunicacão compartilhada. Exclui-se o caso de comunicacão através de memória comum compartihada;

(4) existe algum controle a nivel global, de modo a prover cooperacão dinâmica entre processos e gerenciamento em tempo de execução;

(5) os atrasos sofridos por mensagens trocadas entre processos comunicantes são variáveis e sempre existe algum intervalo de tempo diferente de zero entre a produção de um evento por um processo e a materialização desta produção no ambiente do processo destinatário.

A caracteristica (5) expressa uma restrição fisica importante e fundamental que deve ser levada em conta explicitamente na concepção de sistemas distribuidos. Os sistemas centralizados usam técnicas de controle baseadas na premissa de que é sempre

possivel obter uma visão completa e consistente do estado global do sistema. Nos sistemas distribuídos, por causa da característica (5), pode-se mostrar que é impossível obter uma visão completa e consistente do estado global. A noção de estado global é substituida pela noção de estado local experimentada por cada processo constituinte da parte de controle do sistema. Desse modo, deve ser preocupação constante dos projetistas evitar que decisões de controle se baseiem no estado global, para que não sejam inconsistentes. Estas considerações são extendidas para as aplicações de sistemas distribuídos na automação de processos industriais. Não é possivel , a um processo computacional num ambiente de controle distribuído, dispor de uma visão completa e consistente do processo físico sendo controlado.

2.2.4 Estruturas de Interconexão

Os diversos elementos de um sistema distribuído, por estarem geograficamente separados, necessitam de um meio físico para transportar a informação que trocam entre si. Existe uma diversidade de tais meios físicos, e estes, por sua vez, se estruturam de diversas formas.

A informação é transmitida em forma de sinais elétricos. O meio mais simples de transmissão de sinais elétricos é o par de fios. Este inclui o elementar par trançado e o cabo coaxial. A velocidade útil máxima deste meio, medida em bits por segundo ou em frequência, é imposta pelas propriedades eletromagnéticas da instalação.

Outro tipo de meio utilizado para transmissão é a fibra ótica ou guia de luz. São fibras de material transparente à luz, que têm a propriedade de transmitir sinais luminosos de uma extremidade a outra. Com a adição de elementos convenientes em suas extremidades, é possivel transmitir sinais elétricos. Apresentam caracteristicas de alta velocidade de transmissão e imunidade a ruídos. Têm um futuro promissor em aplicações de controle de

processo.

E' uma maneira também muito usada a propagação de ondas eletromagnéticas, por rádio ou laser. Na maioria das aplicações, as ondas são difundidas no espaço e podem ser captadas livremente por qualquer elemento do sistema distribuído. Este meio é baseado em estações terrestres de rádio ou em satélites espaciais. Este último caso se caracteriza por um atraso elevado na transmissão, da ordem de 0,5 s.

Pode-se ver um sistema distribuído como composto de dois subsistemas: o primeiro constituído dos elementos de computação ou hospedeiros, que são máquinas destinadas à execução de programas do usuário; o segundo é a sub-rede de comunicação ou sistema de interconexão, que conecta os hospedeiros e transporta informação entre eles. O sistema de interconexão, por sua vez, se compõe de meios físicos de transmissão e de nodos ou elementos de chaveamento. Os meios físico são às vezes chamados de linhas de transmissão, circuitos e canais. São implementados pelas tecnologias vistas anteriormente. Os hospedeiros são conectados aos nodos, que por sua vez são interligados via meio físico de transmissão.

Os sistemas de interconexão são basicamente de dois tipos:

(1) sub-rede ponto-a-ponto, e
(2) sub-rede de difusão.

Nas sub-redes ponto-a-ponto, os nodos são interligados aos pares por cabos (fios, fibras óticas) ou mesmo por linhas telefônicas. Se dois nodos não diretamente conectados por um mesmo cabo desejam se comunicar, a transmissão deve ser feita indiretamente através de outros nodos intermediários. A informação, em forma de mensagens, é recebida integralmente por um nodo intermediário que, por sua vez, destina-a a outro nodo. Este pode ser o destinatário final ou ainda intermediário. Uma questão importante nestas sub-redes é a forma de interligação dos nodos, ou seja, a topologia da sub-rede. Existe uma diversidade de possíveis

topologias, algumas das quais representadas na figura 2.1, conforme apresentado por Tanenbaum (1981). As topologias simétricas são usuais em redes locais, enquanto que as redes de longa distância, usando linhas telefônicas, têm em geral topologia irregular.

Nas sub-redes de difusão, o canal de comunicação é compartilhado por todos os nodos. Um deles envia uma mensagem e todos os outros a recebem. Deve haver um mecanismo de forma a indicar a qual ou quais nodos a mensagem é destinada. Os nodos que recebem uma mensagem não destinada a eles, simplesmente a ignoram. Os meios físicos utilizados são os fios e a propagação de ondas eletromagnéticas (radio terrestre ou satélite). A figura 2.2 apresenta alqumas possibilidades para este tipo de sub-rede.

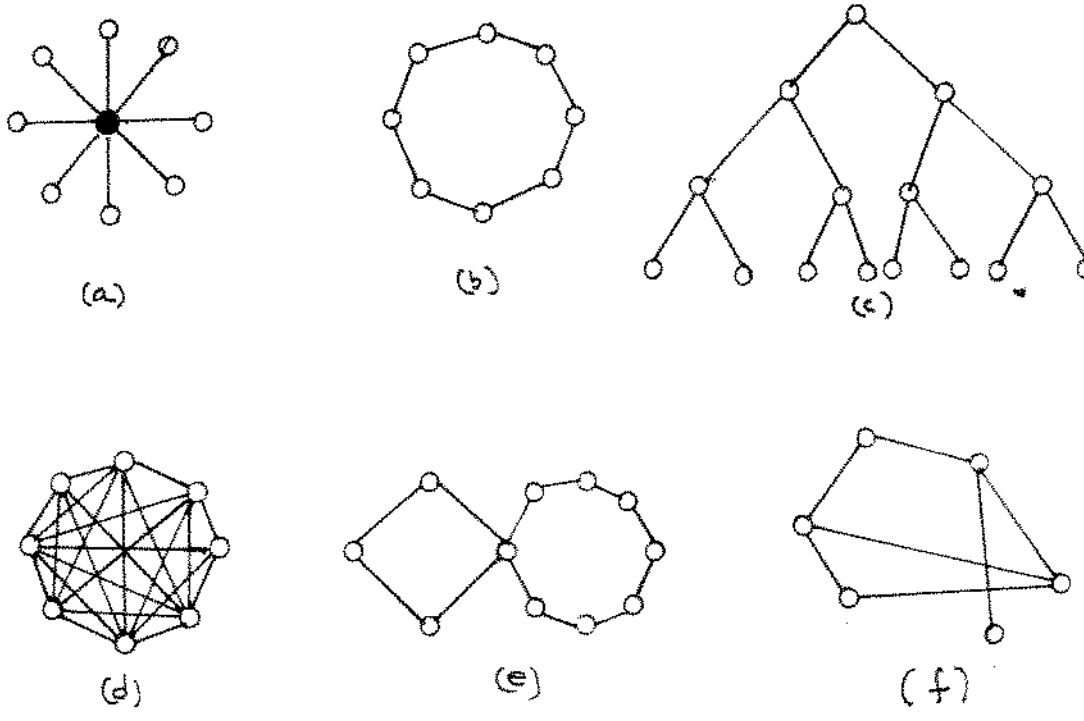


Figura 2.1 Topologias de sub-redes ponto-a-ponto
(a) estrela (b) anel segmentado (c) árvore
(d) completa (e) anéis intercedentes (f) irregular

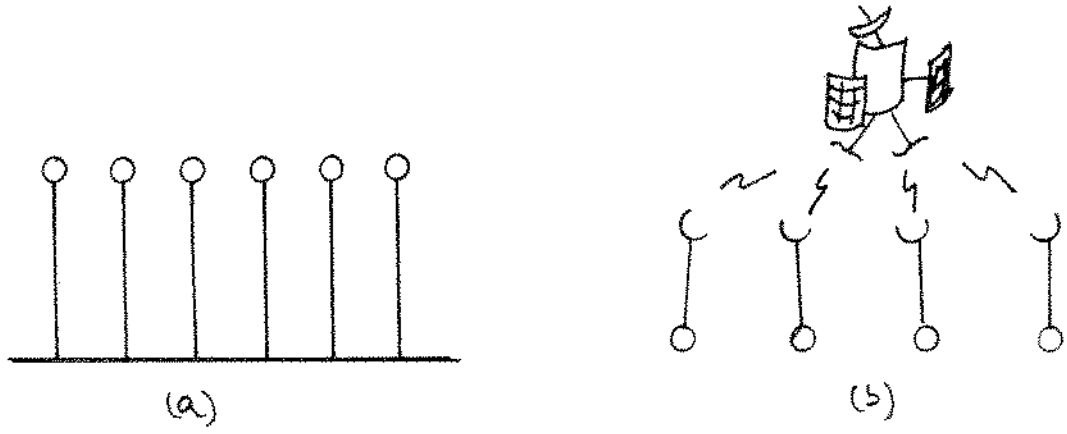


Figura 2.2 Sub-rede de difusão
(a) barramento (b) rádio ou satélite

Nos sistemas de barramento, um nodo que vai enviar uma mensagem deve primeiramente obter permissão para transmitir. O controle do barramento, para resolver conflitos de acesso, pode ser centralizado ou distribuído. Nos sistemas com satélite ou rádio, os nodos devem ser providos de antenas para transmitir e receber.

As estruturas de interconexão foram estudadas por vários autores e existem tentativas de se elaborar taxonomias, entre as quais se destaca o trabalho de Anderson & Jensen (1978). Todavia, não há consenso ainda. As estruturas mais usuais em redes locais serão vistas com mais detalhes em seção posterior.

2.2.5 Arquitetura dos Sistemas Distribuídos

Os sistemas distribuídos são de natureza em geral complexa, tanto pela parte referente às aplicações, como pela parte referente à comunicação entre os componentes distribuídos e separados fisicamente. Contribuem para esta complexidade questões como a

troca de informacão à distância e o controle da sincronização e da consistência das operacões desempenhadas em diferentes regiões (Bochmann, 1983).

A preocupacão de se dominar a complexidade dos sistemas de computacão já vem de algum tempo. No que tange principalmente à concepcão de programas, uma série de métodos, critérios, técnicas e ferramentas tem surgido e, coletivamente, caracterizam a disciplina chamada "Engenharia de Software". Uma das idéias básicas nesta disciplina consiste na busca de soluções que sejam estruturadas em módulos, aqui usados no sentido intuitivo. Os módulos devem apresentar a maior independência possível um dos outros. Em sistemas distribuídos a mesma idéia se aplica. Sejam dois ou mais computadores interconectados fisicamente e cada um deles organizado como uma máqina composta de uma série de camadas ou níveis de abstração. Estes vão desde o nível de "hardware", passando pelos níveis de linguagem de máquina, de sistema operacional, de linguagem de montagem, de línguagens de alto-nível, até o nível de linguagens orientadas para problemas. Acima de tudo pode-se ver uma coleção de programas afetos a aplicações particulares, coletivamente chamados de aplicação. O problema de sistemas distribuídos consiste em permitir e facilitar a troca de informação em ambientes deste tipo. As soluções para estes problemas estão sendo organizadas, cada vez mais, em módulos estruturados de forma a refletir a estrutura do problema. Assim, todo suporte à comunicação é estruturado em camadas, cada uma tendo uma função e executando certos serviços. A fronteira entre as camadas deve ser nítida. No ítem seguinte será visto a estruturação dos sistemas distribuídos.

2.2.5.1 Camadas, entidades, protocolos e interfaces

Dois processos computacionais comunicantes de maneira direta, sem nenhuma interferência de terceiros, podem ser esquematizados como na figura 2.3 (a). O esquema pode ser generalizado para um número

qualquer de processos. Todavia, como acontece em grande parte dos casos, esta comunicação não pode ser feita de maneira direta. Por exemplo, quando cada processo está situado em um computador distinto. A troca de informação se dá então através de um recurso específico para suportar a comunicação entre os processos. Este se situa entre os dois processos comunicantes, como mostra a figura 2.3 (b). Sua função é receber informação de um dos processos e transportá-la para o outro, de maneira a tornar transparente a ambos a existência do que quer que esteja no caminho. Vê-se aí uma estruturação hierarquizada do sistema, composto por processos nas extremidades e pelo suporte à comunicação no meio. E' usual representar a estrutura dispondo verticalmente as partes do sistema, como mostrado na figura 2.3 (c). O suporte à comunicação pode, por sua vez, ser também hierarquizado, como mostra a figura 2.3 (d).

Cada nível da hierarquia constitui uma camada de serviço. Associado com uma camada existe uma interface com a camada superior e outra com a inferior. Uma camada de um nível n é referenciada como camada-(n). O objetivo de uma camada-(n) é prover certos e bem definidos serviços para a camada-(n+1) e superiores, escondendo os detalhes de como o serviço oferecido é implementado. A camada-(n) é implementada baseando-se nos serviços oferecidos pela camada-(n-1). A figura 2.4 ilustra estes e outros pontos a serem tratados a seguir.

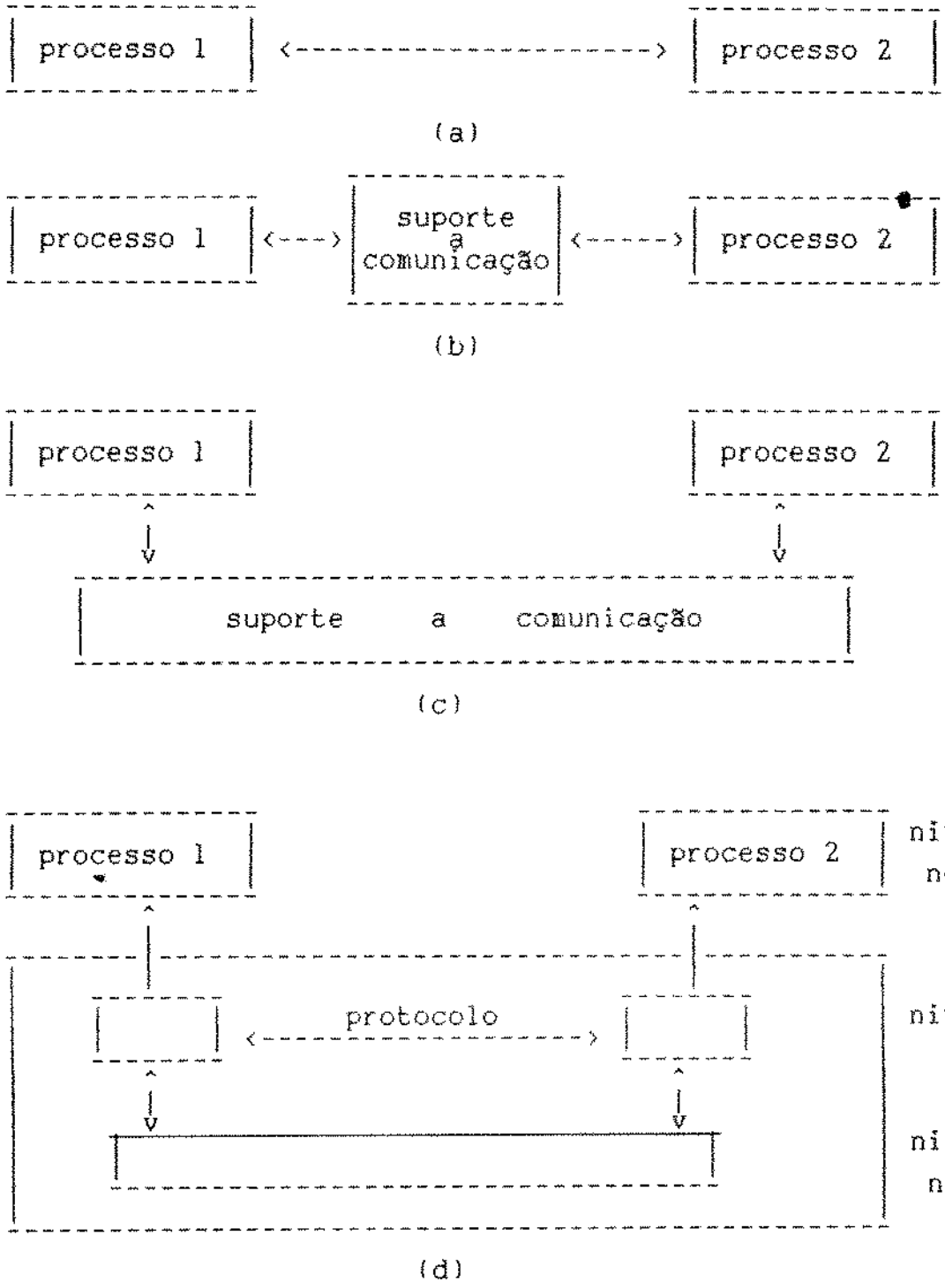


Figura 2.3 Principio de estruturação em camadas

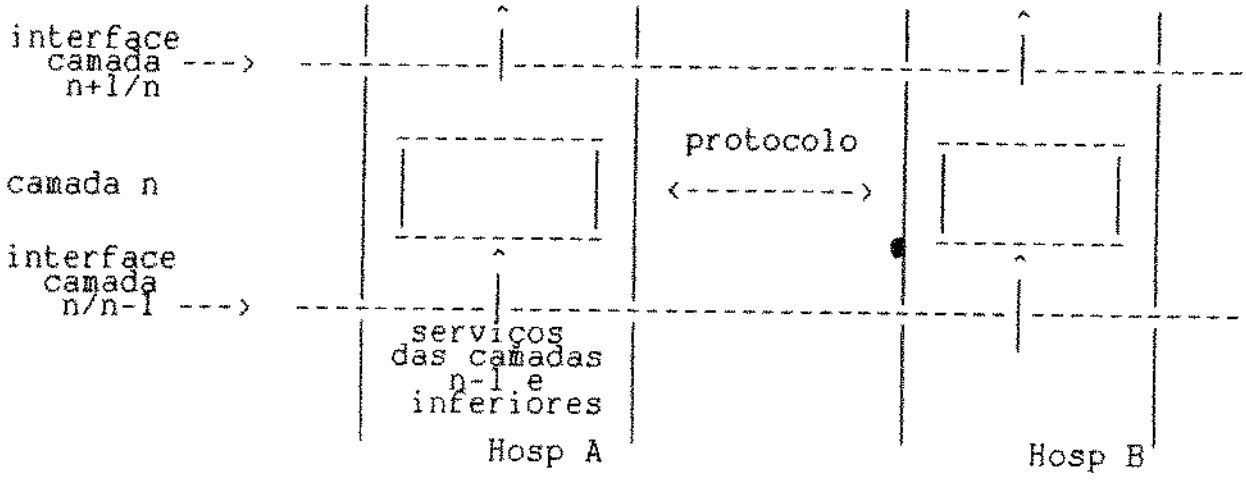


Figura 2.4 Camadas, protocolos e interfaces

Uma camada se constitui de componentes chamados de entidades, implementados em cada elemento da rede, seja um nó ou um computador hospedeiro. Para executar seus serviços as entidades de uma camada mantêm uma conversação. Existe um conjunto de regras e convenções usadas na conversação entre entidades de uma camada que são coletivamente conhecidas como protocolo da camada-(n).

Os serviços da camada-(n) oferecidos à camada-(n+1) e superiores são providos através de uma interface. Esta define, por meio de operações primitivas, quais são os serviços oferecidos e de que maneira os mesmos são alcançados.

A transferência de dados entre entidades de uma mesma camada não se dá diretamente. As informações (de dado e de controle) são na realidade passadas para a camada imediatamente abaixo, até chegarem à camada mais baixa, onde se situa o meio físico de comunicação. Aí ocorre ocorre a transferência real de informação entre suas entidades afastadas geograficamente. A figura 2.5 ilustra estes aspectos, com um exemplo de 4 camadas. As linhas pontilhadas, representando os protocolos, indicam a existência de uma comunicação virtual. A linha contínua indica que há comunicação real ou física através do meio físico de interconexão.

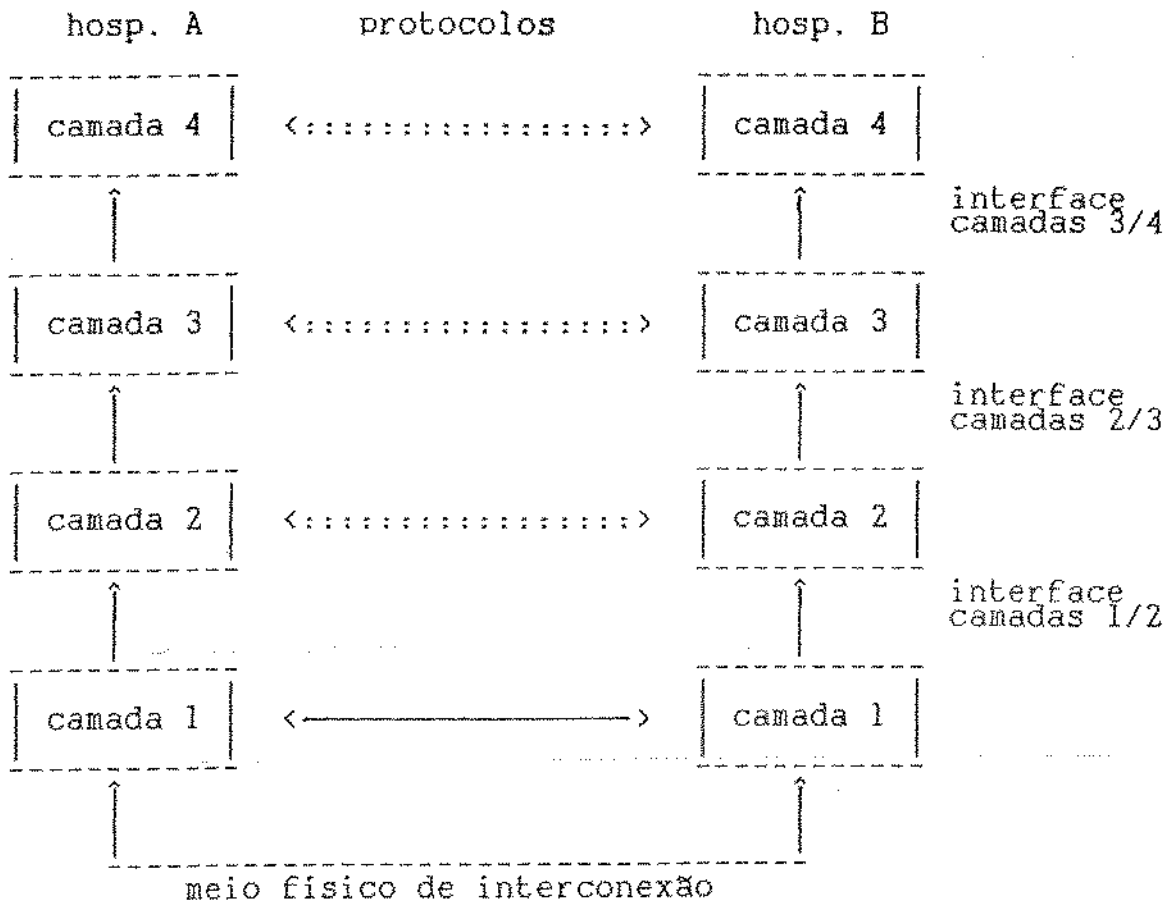


Figura 2.5 Exemplo de Arquitetura Multicamadas

Ao conjunto de camadas, protocolos e interfaces dá-se o nome de arquitetura do sistema de interconexão. Deve-se notar que a composição da arquitetura de sistemas distribuídos leva em conta somente aspectos relativos à comunicação entre os hospedeiros. Outros aspectos pertinentes a cada hospedeiro em particular, como por exemplo, arquitetura e sistema operacional local, não são considerados.

2.2.5.2 O modelo de arquitetura da ISO

A organização da arquitetura em camadas hierarquizadas apresenta uma série de vantagens, que estão relacionadas, por exemplo, em Watson (1981). O processo de decomposição do sistema em camadas é iterativo e um tanto arbitrário. Existem porém, na literatura, orientações para especificação das camadas (Zimmermann, 1980). A par de tudo isso é de reconhecida importância a necessidade de estabelecimento de um padrão para a arquitetura de sistemas de interconexão. Uma das grandes características de sistemas distribuídos consiste na necessidade frequente de interconexão de computadores heterogêneos, inclusive de fabricantes diferentes. Para que um computador faça parte do sistema distribuído, ele deve seguir os mesmos protocolos seguidos pelos outros. Esta preocupação vem de algum tempo e existe um esforço internacional bem sucedido de padronização da arquitetura. O estágio atual da padronização está mostrado em Rosemberg (1984). Um padrão foi, e ainda está sendo, desenvolvido pela ISO (International Organization for Standardization), um organismo internacional que congrega entidades de vários paises interessados em aspectos de padronização.

A ISO elaborou um modelo básico de referência da arquitetura de sistemas de interconexão, referido como Modelo OSI - "Open Systems Interconection" (ISO, 1983). Este documento provê uma base comum para a coordenação do desenvolvimento de padrões. A arquitetura é composta de sete camadas, as quais são descritas aqui resumidamente. Cada camada particular é objeto de estudos para normalização. Algumas já foram totalmente padronizadas e outras estão em processo de padronização. As camadas, sucintamente descritas abaixo, são mostradas em conjunto na figura 2.6.

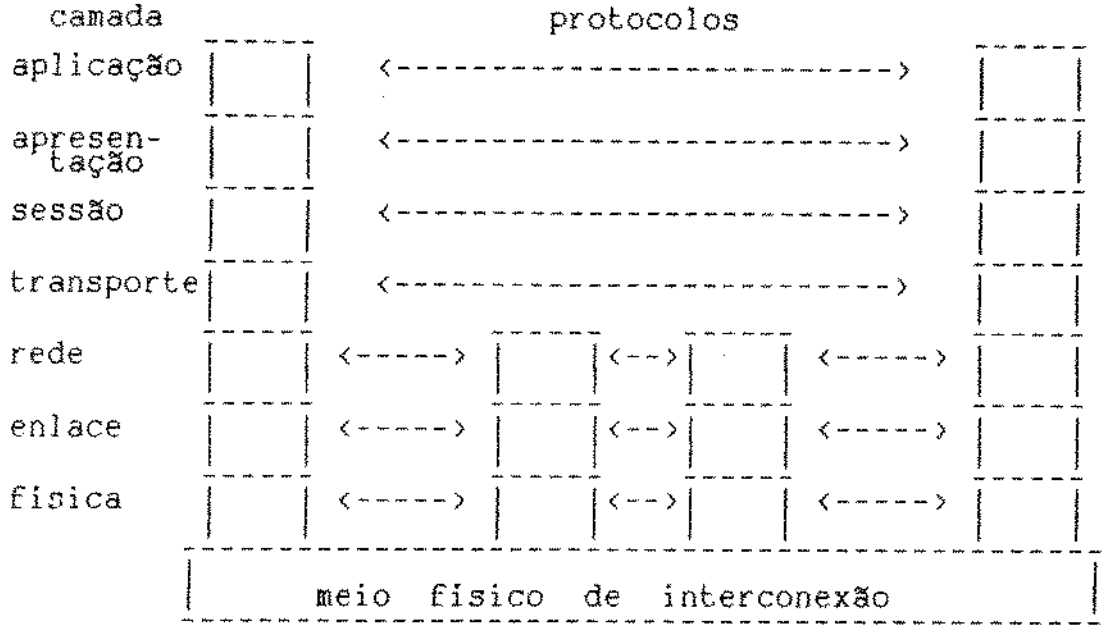


Figura 2.6 As sete camadas da arquitetura OSI

Camada de aplicação

Está no nível mais alto do modelo básico OSI. Provê diretamente serviços para os processos dos usuários finais. Os protocolos nesta camada visam atender às aplicações dos usuários, ao gerenciamento destas aplicações e ao gerenciamento do sistema nas atividades de comunicação. Nessa camada os usuários têm a liberdade de definir seus próprios protocolos. Os processos da aplicação são os consumidores finais e geradores primários de informação. Os processos dessa camada podem ser de qualquer tipo - manual, computadorizado, industrial e físico. Ao lado de protocolos específicos para cada aplicação, podem existir protocolos que atendam a uma gama de aplicações, por exemplo, protocolos de transferência de arquivos ou de banco de dados distribuídos. Esta camada apresenta o maior número de questões ainda em aberto.

Camada de apresentação

Objetiva a execução de funções frequentemente requisitadas e que por isso podem ser resolvidas de forma geral ou genérica, ao invés de deixá-las a cargo de cada usuário. Um problema típico é a conversão de códigos de caracteres: por exemplo, o estabelecimento de um código comum a ser usado na transferência de informação e a conversão, para compatibilização de hospedeiros heterogêneos quanto à forma de representação de dados. Outro problema é a manutenção da segurança da informação, resolvido possivelmente com o uso de criptografia. A compressão de dados é outra função requisitada nesta camada.

Camada de sessão

Tem como objetivo fornecer os meios necessários para organizar e sincronizar o diálogo entre entidades cooperantes da aplicação. A camada, para isto, estabelece uma relação de cooperação entre as entidades, chamada de "sessão". No estabelecimento da sessão, devem ser considerados pontos como endereçamento e autenticação dos cooperantes. Também deve haver concordância sobre uma variedade de opções que serão ou não levadas a efeito durante a sessão - determinação de um conjunto único de valores de parâmetros de operação. P. ex., se a comunicação vai ser bidirecional plena ou alternada ("full duplex" ou "half duplex"). Durante a existência de uma sessão é provida uma troca ordenada de mensagens e também um diálogo regulado entre as entidades cooperantes.

Os serviços oferecidos pelas camadas de aplicação e sessão podem ou não ser necessários, como consequência das características particulares de sistemas distribuídos e aplicações. Exemplificando, num sistema distribuído em que os computadores usam o mesmo código de representação de dados, a conversão é desnecessária. Em certas aplicações, nas quais se encontra a de controle de processos industrias, muitas vezes não se requer o estabelecimento de uma conexão para que duas entidades possam comunicar entre si: os parâmetros são conhecidos "a priori" e todos os elementos necessários para a comunicação, inclusive os dados, estão contidos inteiramente numa mensagem.

Desse modo, a inclusão ou não dos serviços de apresentação e sessão na arquitetura de um sistema distribuído em particular, vai depender de suas características.

Camada de transporte

Esta camada provê para as superiores um serviço independente do meio de comunicação (camada inferior), de forma a tornar transparente a existência da sub-rede de comunicação, sua estrutura e características tecnológicas. Deve garantir a transferência correta, ordenada e de maneira eficiente, da informação gerada em um elemento de computação até seu destino final, usando apropriadamente os serviços da camada inferior. Os protocolos desta camada têm significado fim-a-fim, ou seja, o diálogo se dá entre duas entidades terminais, uma sendo fonte primária e outra consumidora final. Percorrendo as camadas no sentido de cima para baixo, esta é a última camada em que ocorre tal tipo de comunicação, como pode-se ver na figura 2.6. Os seus serviços são:

estabelecimento e encerramento de conexões de transporte;

provimento de classes variadas e selecionáveis de serviço. Estas classes devem cobrir as diferentes necessidades de tráfego geradas pelas camadas superiores;

transferência de dados com qualidade solicitada.

Entre as funções executadas para produzir o serviço desejado, tem-se:

mapeamento de endereço a nível de transporte para endereços a nível de rede (a camada inferior);

multiplexação e demultiplexação de conexões;

segmentação de mensagens em unidades menores, reagrupamento e consequente controle de sequência;

controle do fluxo de informação;

detecção e correção de erros.

Camada de rede

Esta camada está afeta à transferência de informação entre os diversos componentes da sub-rede de comunicação. A unidade de informação tratada é conhecida como pacote. A comunicação entre as entidades desta camada é levada a cabo de maneira ponto-a-ponto. Isto é, uma entidade envia dados para outra intermediária, situada em um nodo vizinho; esta por sua vez envia os dados para uma terceira; o processo se repete até que os dados cheguem ao seu destino final. A figura 2.6 ilustra estes aspectos. Isto gera a necessidade do estabelecimento de rotas para os dados. Entre os serviços e funções estão o controle de fluxo, para evitar congestionamentos, o sequenciamento, detecção e correção de erros de pacotes, e multiplexação de conexões de rede para otimização de recursos de comunicação. E' nesta camada que se estabelece uma interface entre cada hospedeiro e a sub-rede de comunicação. A partir desta camada, torna-se relevante a tecnologia e estrutura da sub-rede de comunicação. A propósito, foi recentemente implantada no Brasil, pela Embratel, uma sub-rede pública usando técnicas de comutação de pacotes, a RENPAC (Embratel, 1984).

Camada de enlace de dados

O objetivo desta camada é prover, para duas entidades comunicantes da camada de rede, uma linha de comunicação que se aparente livre de erros, a partir da linha física existente. Os dados são segmentados e encaixados em unidades conhecidas como quadros. E' sua função delimitar os quadros e transmiti-los em sequência. Quando um quadro é recebido, o receptor envia outro de reconhecimento. Um conjunto de erros pode surgir nesta

camada: perda, duplicação e erro de transmissão de quadros. Devem ser providos mecanismos para tratar tais situações, de forma a manter a aparência de linha livre de erros. Além disso, deve haver controle sobre o fluxo de dados enviados/recebidos.

Camada física

Responsável pela transferência de bits através do meio físico de transmissão. A unidade tratada nesta camada é o "bit". Relaciona-se com as características elétricas e mecânicas da transmissão. São usuais questões do tipo quantos volts representam o bit "1" e quantos o bit "Ø", qual a duração dos estados que representam os bits "1" e "Ø", qual (ou quais) o tipo de conector que liga o nodo à linha física, etc. E' função também desta camada o estabelecimento de conexões físicas, o sequenciamento dos bits, e notificação de condições de erro para a camada de enlace.

A padronização de protocolos deste modelo de referência alcança já, em termos de ISO, um estágio relativamente avançado. Assim, existem propostas, bem aceitas, para padronização dos níveis físicos, de enlace, de rede e de transporte. Nos níveis superiores há uma grande atividade neste sentido. Para as redes locais, que têm características especiais, há também iniciativas avançadas de padronização (IEEE, 1983).

2.2.6 Serviços de Datagramas e Circuitos Virtuais

Define-se como sub-rede de comunicação a porção da rede constituida das camadas física, de enlace e de rede. Dois tipos de serviço são usualmente prestados pela sub-rede, bem como pela camada de transporte: de datagramas e circuitos virtuais.

No serviço de datagrama a unidade de informação é chamada de "datagrama" que, para a camada de rede, corresponde ao "pacote" e, para a camada de transporte, corresponde à "mensagem". Os datagramas são enviados independentemente uns dos outros. Não há qualquer ordenação entre os datagramas e não há garantia de entrega ao destinatário. Cada datagrama deve conter os endereços de origem e de destino. Ao receber um datagrama, no caso de sub-rede, um nó consulta uma tabela interna de endereços para identificar qual o caminho a ser tomado pelo datagrama.

No serviço de circuitos virtuais é oferecido um canal virtual de comunicação entre duas entidades comunicantes. A ordem de transmissão de pacotes ou mensagens é mantida e, possivelmente, é garantida a entrega ao destinatário. O primeiro pacote ou mensagem carrega o número do circuito virtual sendo estabelecido e o endereço de destino. Em cada nó vai se estabelecendo o caminho entre a origem e o destino, até o nó que contém o destinatário. Os próximos pacotes carregam apenas o número do circuito virtual, além dos dados.

Esta seção tratou de aspectos relacionados com sistemas distribuídos de maneira geral, apresentando os conceitos e alguns problemas envolvidos. Embora este trabalho se relacione mais com redes locais, o tratamento inicial generalizado se justifica pelas seguintes razões, entre outras:

- é importante ter-se uma visão mais ampla do assunto, no sentido de melhor se situar quando da concepção de sistemas específicos;

- a problemática da comunicação em redes locais é semelhante à

dos sistemas distribuídos de maneira geral;

- certas aplicações de redes locais tendem a gerar necessidades de serviços semelhantes aos serviços usuais em redes gerais.

2.3 REDES LOCAIS DE COMUNICAÇÃO

Continuando a abordagem no sentido geral para o especifico, apresenta-se nesta seção as redes locais de comunicação. Estas são essenciais em sistemas distribuidos, segundo a caracterização anteriormente feita, os quais, entre outras aplicações, se prestam ao controle de processos e à automação da manufatura discreta. Assim, serão apresentados os conceitos envolvidos em redes locais de maneira geral.

2.3.1 Conceitos, Topologias e Meios de Transmissão

Uma rede local de comunicação, segundo definição de Sttalings (84), é uma rede de comunicação que objetiva a interligação de uma variedade de dispositivos de comunicação de dados. O termo rede de comunicação é aqui usado no sentido diferente do termo rede de computadores, que é composta de elementos de interligação e de computadores hospedeiros, além do software de implementação de protocolos de niveis superiores. Por dispositivos de comunicação de dados entende-se qualquer dispositivo que se comunique com outros via um meio de transmissão de dados. Entre os dispositivos citam-se os terminais inteligentes, os computadores, os dispositivos convencionais de entrada/saida, os telefones, e os sensores transmissores/receptores de televisão.

As redes locais se prestam potencialmente a uma variedade de classes de serviços. tais como transmissão de arquivos, processamento remoto, monitoração e operação remota de equipamentos e processos, transmissão de voz e videotexto. As três últimas classes de serviço caracterizam-se, muitas vezes, por exigirem comunicação em tempo real e por perfis estatísticos de tráfego diferentes daqueles dos serviços convencionais.

Numa rede local de comunicação os seus componentes estão situados em uma pequena área - de um edifício, de um campus, de uma

fábrica ou até de uma cidade. Isto traz uma série de consequências em razão da curta distância entre os componentes. Além do mais, uma rede local é, em geral, propriedade de uma organização ou de um grupo com objetivos afins. Isto favorece o uso de soluções mais especificas que podem, portanto, ser mais simples e eficientes.

O custo de uma rede local é altamente variável, dependente das características das aplicações às quais se prestam. Os três fatores relevantes na composição do custo são a velocidade exigida de transmissão, a confiabilidade da comunicação e o grau de generalidade de acoplamento de dispositivos.

As redes locais de comunicação se caracterizam tanto pela generalidade de usuários e dispositivos que podem ser conectados, como pela padronização da cooperação entre seus componentes. A maioria das redes locais são concebidas para aplicações especificas, por exemplo, as redes para implementação de sistemas digitais de controle distribuido. Todavia é uma tendência recente o projeto de redes locais para integração de serviços de variadas naturezas (transmissão de voz, transmissão de imagem, controle de processos, videotexto e processamento convencional). As redes para integrar esses serviços são conhecidas como redes com serviços integrados (Zuchi & Ruggiero, 1982).

Neste ponto de vista, a titulo de ilustração, a automação da manufatura vai demandar redes com serviços integrados. A automação da manufatura envolve as seguintes classes de funções:

- de projeto (CAD - "Computer Aided Design" e CAE - "Computer Aided Engeneering");

- na manufatura propriamente dita (robôs, máquinas ferramentas a CNC - "Computerized Numerical Control", sistemas de controle automático de transporte, manuseio e armazenamento, etc);

- na administração e planejamento de produção (CAP - "Computer Aided Production" e CAPP - "Computer Aided Process

Planning").

A integração e harmonização dessas funções caracteriza o que é chamado de CIM - "Computer Integrated Manufacturing". Essas funções demandam diferentes tipos de serviços de transmissão de dados. A interconexão dos diversos dispositivos para implementar sistemas CIM deve se dar de forma natural com as redes locais.

Os valores usuais para redes locais tipicamente são os seguintes:

. taxas de transferência de dados altas (0,1 a 100 Mbps)

. distâncias relativamentes curtas (0,1 a 50 km)

. taxas de erro baixas (10^{-8} a 10^{-11})

Os meios físicos de transmissão de dados, para interconexão dos componentes são de 3 tipos:

. par trançado de fios

. cabo coaxial

. fibra ótica

O par trançado é o mais comum, usual para baixas velocidades (até 1 ou 2 Mbps) e distância de poucos quilômetros. Permite conexão de poucos dispositivos (dezenas). E' próprio para transmissão de sinais digitais. Tem uma considerável susceptibilidade a ruídos e interferências, porém apresenta um baixo custo.

O cabo coaxial apresenta um melhor desempenho, com taxas máximas de transmissão de 10 a 50 Mbps; alcança maiores distâncias; permite um número mais elevado de dispositivos conectados. Existem os cabos coaxiais de 50 ohm e de 75 ohm. O primeiro é para transmissão digital de sinais e o segundo serve tanto para transmissão digital como analógica.

A fibra ótica é um meio de futuro promissor em redes locais, apresentando taxas de transmissão da ordem de 10 a 100 Mbps;

permite distâncias curtas (até 1 km); o número prático de dispositivos conectados é da ordem de dezenas; tem uma baixa susceptibilidade a ruídos; o custo ainda é alto, mas da mesma ordem do custo dos cabos coaxiais. Podem ser usadas tanto em configuração ponto-a-ponto como multiponto. Neste último caso existem ainda limitações técnicas decorrentes do uso de "taps" de acesso: perdas e reflexões óticas.

A natureza de uma rede local é determinada principalmente pelo esquema de interconexão dos seus componentes (topologia) e pelo meio físico de transmissão. Basicamente são três os tipos de topologias usuais (redes de topologias simétricas):

. estrela

. anel

. barramento

Na topologia em estrela, existe um elemento central chaveador que conecta todos os componentes do sistema. O desejo de uma estação ou componente de se comunicar com outra é levado ao elemento chaveador através de uma requisição. Este estabelece um caminho dedicado entre as estações. A comunicação passa então a ser do tipo ponto-a-ponto. E' um esquema de implementação simples, porém apresenta a grande desvantagem de se basear em um elemento ativo centralizador, com consequências prejudiciais à característica desejável de confiabilidade dos sistemas distribuídos.

Na topologia em anel, cada estação é ligada a uma interface de comunicação, que se liga com outras num esquema ponto-a-ponto em forma de um laço fechado. Os dados circulam no laço passando serialmente pelas interfaces. Uma estação que deseja transmitir coloca os dados em forma de mensagem no anel. A mensagem deve conter os endereços da estação fonte e destino. A interface associada à estação destino se encarrega de copiar a mensagem que chega e lhe é destinada. Usualmente a mensagem copiada continua circulando no laço até voltar à interface de origem, que a retira de circulação, provendo assim uma forma de confirmação. Qualquer um dos três meios de transmissão podem ser usados nesta

topologia. O controle de acesso ao meio nesta e na topologia de barramento será visto adiante. Exemplos de redes com este esquema são a Rede de Cambridge (Needham, 1979) e a rede DLCN (Liu, 1978). Um estudo detalhado sobre redes em anel se encontra em Penney e Bagdadi (1979a e 1979b).

Na topologia de barramento, existe um meio único de transmissão, com acesso múltiplo e compartilhado por todas as estações. Os dados são depositados no meio e todas as outras estações têm acesso simultâneo a eles. Uma estação transmite de cada vez, colocando no meio mensagens que incluem identificação da fonte e destino. E' uma topologia propicia para ocorrência de conflitos de acesso ao meio. Os meios físicos usados são o par trançado e o cabo coaxial. A fibra ótica, por apresentar problemas na ligação multiponto, é utilizada apenas em função de pesquisas. A rede em barramento mais conhecida é a Ethernet (Metcalfe e Boggs, 1976).

2.3.2 Protocolos de Controle de Acesso ao Meio

O meio de transmissão, sendo um recurso compartilhado pelas estações componentes da rede, deve ter seu acesso e utilização de maneira disciplinada. Para cada tipo de rede existe um conjunto de regras e técnicas conhecidas como protocolos de controle de acesso. Uma questão a ser resolvida no controle de acesso é "onde" se fazer este controle. Dois enfoques opostos são utilizados: centralizado e distribuído. O controle centralizado em um único local da rede traz algumas vantagens, entre elas a possibilidade de se obter um maior controle sobre o acesso e uma maior simplicidade na lógica das estações. Porém este enfoque traz consequências negativas para a disponibilidade, devido à existência de um ponto crítico sujeito a falhas. Ademais, este ponto funciona como um gargalo, podendo reduzir a eficiência.

O enfoque distribuído, onde as decisões são tomadas nas estações, traz problemas de coordenação e controle do acesso e de complexidade na lógica das estações. Todavia é favorável ao objetivo disponibilidade e permite uma utilização mais eficiente

do recurso.

O meio, para ser utilizado deve ser multiplexado. A multiplexação pode ser por divisão de frequência, e, de maior interesse, por divisão de tempo. O acesso, neste caso, pode ser aleatório ou controlado. As técnicas usuais de acesso aleatório são sucintamente descritas abaixo.

CSMA - "Carrier Sense Multiple Access"

Usada em redes de barramento. Todas estações "escutam" a linha e, dependendo se ela está sendo usada, podem ou não transmitir. Quando uma estação deseja transmitir e a linha está ocupada, ela espera por um período de tempo, segundo algum algoritmo, após o que tenta novamente. Quando uma mensagem é transmitida a estação espera por uma confirmação do destinatário. No caso da transmissão não ser bem sucedida, a estação retransmite a mensagem.

CSMA/CD "Carrier Sense Multiple Access with Collision Detection"

Também para redes em barramento. Neste caso, e estação escuta a linha enquanto envia sua mensagem. Se ocorrer uma colisão de mensagens na linha, a estação interrompe a transmissão e, após um intervalo aleatório, tenta novamente.

Anel Ranhurado

Uma quantidade de ranhuras (espaço para mensagens) de tamanho fixo circula no anel. Cada ranhura é marcada se vazia ou não. Uma estação que deseja transmitir espera por uma ranhura vazia, na qual insere a mensagem de dados. Esta circula no anel, é lida pelo destinatário, o qual altera um campo de confirmação. Quando a mensagem retorna à fonte ela é retirada ou retransmitida, em

função do campo confirmação. É um esquema de muita simplicidade.

Inserção de Registro

Usado em anel, é uma versão mais elaborada do anel ranhurado. Cada estação tem um registrador deslocador de tamanho do máximo pacote que circula, utilizado para armazenar temporariamente os pacotes de dados circulantes. Tem também um registrador para pacotes originados na estação. As estação que deseja transmitir um pacote o faz colocando-o no registrador deslocador, se houverem posições consecutivas suficientes. A retirada da mensagem do anel pode ser feita ou pela estação destinatária ou pela emitente. A maior vantagem deste método é que ele permite a máxima utilização do anel, se comparado com os outros.

Na técnica de acesso controlado, o momento e a sequência de acesso são regulados por algoritmos. Duas delas seguem:

Passagem de Permissão em Barramento

Existe uma ordenação de estações para adquirir o direito de acesso, formando um anel lógico. Quando uma estação recebe a permissão, em forma de um pacote de controle, ou ela adquire o controle do barramento, fazendo as transmissões necessárias, ou passa a permissão para a próxima estação na sequência. Cada estação deve conhecer sua predecessora e sucessora. Neste método é necessário prever e tratar especificamente algumas situações: inserção e remoção de estações no anel lógico; gerenciamento de falhas do tipo endereço duplicado e ruptura do anel; (re)inicialização do anel; perda do pacote de permissão.

Passagem de Permissão em Anel

Neste método um pequeno pacote de permissão circula pelo anel, indicando que não há estação transmitindo. Uma estação que deseja transmitir deve esperar pela permissão. Quando ela chega, a estação altera o pacote de permissão, para indicar ocupação, e insere-o no anel seguido de um pacote de dados. A estação destinatária copia o pacote de dados. Ao retornar à fonte, o pacote de dados é retirado e o de permissão é restaurado e transmitido. Os principais problemas neste método, causados por erros, são a perda do pacote de permissão e a existência de pacote de ocupação circulante sem pacote de dados associado. Este método permite que o tráfego seja regulado e que se use prioridades.

2.3.3 Padronização em Redes Locais

A definição de um padrão para redes locais, no que concerne ao controle de acesso, é uma necessidade que está concretizando-se. Os maiores interessados na padronização são os usuários e a indústria de semicondutores. Os usuários, por ser indesejável depender de um único fabricante de redes e equipamentos. A indústria de semicondutores, por não ser interessante diversificar demasiadamente a produção de interfaces para ligar equipamentos diferentes. Ademais, com a padronização, o custo de conexão de equipamentos às redes tende a cair, de forma que a conexão saia mais barata que o equipamento conectado.

Dos esforços de padronização, o de maior relevância e sucesso está sendo desempenhado pelo Comitê 802 do IEEE (IEEE, 1983; Myers, 1982). Os vários métodos de acesso vistos têm vantagens e desvantagens relativas, bem como cada um tem aplicações a que se adequam melhor. Por outro lado, fabricantes expressivos têm interesse em que suas redes sejam contempladas na padronização. Como consequência, o Comitê 802 elaborou um padrão, que na realidade se compõe de algumas alternativas padrão. Aquele define uma arquitetura em 3 camadas (ver figura 2.7):

. de controle de enlace lógico;

. de controle de acesso ao meio, e

. fisica.

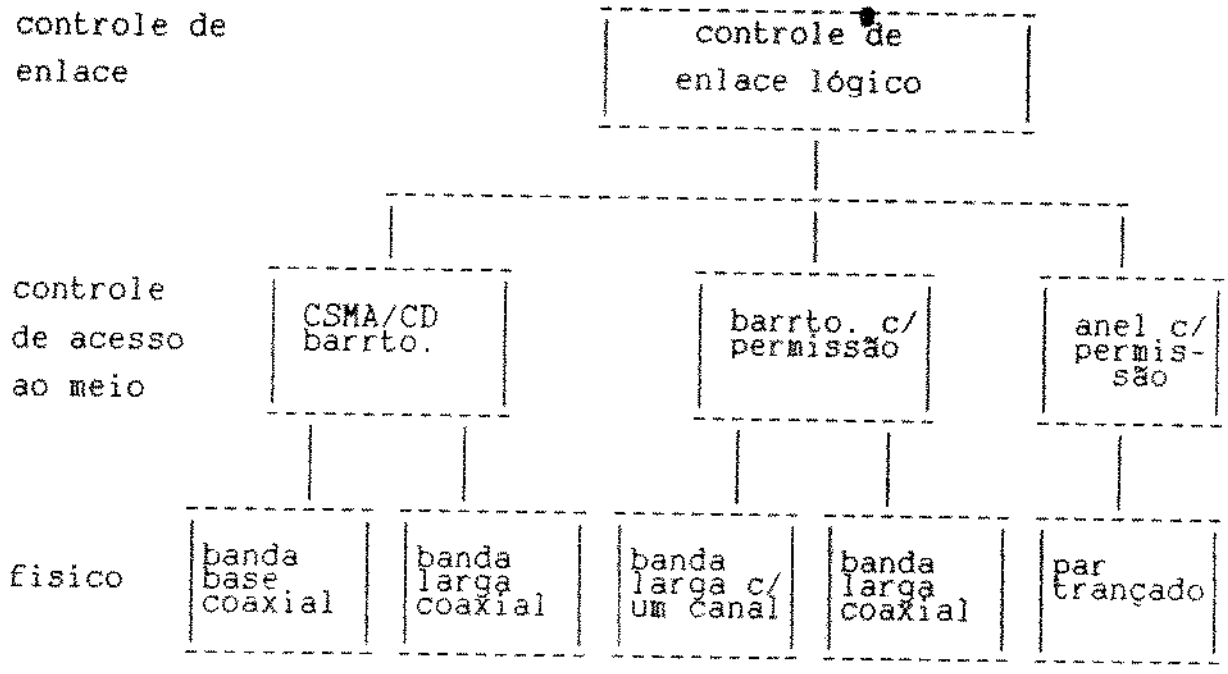


Figura 2.7 O padrão IEEE 802 para redes locais

Se comparada com a arquitetura da ISO para sistemas abertos, vê-se que as três camadas do padrão correspondem às camadas fisica e de enlace da ISO. A camada de enlace lógico provê a troca de informação entre entidades comunicantes do nivel superior, multiplexando o recurso meio fisico de transmissão. Oferece serviços de datagrama, no modo sem conexão, e de circuito virtual, no modo com conexão. Os protocolos usados são semelhantes ao HDLC.

A camada de controle de acesso ao meio tem 3 opções, a saber:

. barramento com CSMA/CD;

. barramento com passagem de permissão;

. anel com passagem de permissão.

À opção de barramento com CSMA/CD coincide com a especificação da rede Ethernet e tem maior aplicação em automação de escritórios. Às outras duas opções, seguindo a idéia de passagem de permissão em barramento e em anel, contemplam aplicações criticas no tempo, onde o tempo de resposta é o fator mais relevante. Nesta classe de aplicações se enquadra o controle de processos em tempo real.

No nivel fisico o leque se abre mais, não havendo sido ainda totalmente fechado. Para CSMA/CD foi proposto o uso de cabo coaxial banda base, com taxa de 10 Mbps, e estão em estudo algumas opções com cabo coaxial banda larga, com propostas de taxas de 1 a 5 Mbps por canal. Para barramento com passagem de permissão, existe a transmissão em banda larga de canal único e taxa de 1 Mbps. Outras taxas, de 5 ou 10 Mbps estão também sendo estudadas. Ainda, foi proposto o uso de cabo coaxial banda larga com taxas de 1, 5 ou 10 Mbps. Finalmente, para o anel, proprô-se o uso de par trançado com taxas de 1 e 4 Mbps.

A escolha desses "padrões" aparentemente pode não ter resolvido muito. Todavia, como é a expectativa do Comitê 802, o leque de opções poderá ser diminuido de forma natural, devido à predominância em termos de uso de alguns métodos sobre outros.

2.4 SISTEMAS DIGITAIS DE CONTROLE DISTRIBUIDO EM CONTROLE DE PROCESSOS

2.4.1 O Computador no Controle de Processos

Com o advento dos computadores digitais, criou-se uma expectativa em torno de sua aplicação em controle de processos. Até então o controle era feito através de controladores analógicos. Inicialmente os computadores digitais foram usados para calcular valores de referência para estes controladores analógicos. O segundo passo foi o uso de computadores em substituição aos controladores convencionais - Controle Digital Direto (DDC, "Direct Digital Control"). Cada computador substituía um conjunto de controladores, executando em sequência o controle dos laços de realimentação. A quantidade de laços controlados era função da capacidade dos computadores, inicialmente baixa. Esta opção abria a possibilidade da aplicação de algoritmos em software elaborados. Porém, trazia problemas de confiabilidade, por ser baseado em um único equipamento. A solução muitas vezes adotada era o uso de controladores convencionais de reserva, para substituição em caso de falha do computador. Este fato, aliado ao alto custo e baixa capacidade dos computadores, levava a soluções de alto custo total. Ademais, o software disponivel para controle ainda deixava a desejar. Desse modo, o controle de plantas industriais complexas não era de todo viável.

Numa terceira etapa, surgiram os computadores de grande porte executando o controle de forma centralizada. A expectativa era de que, devido à grande capacidade de memória, velocidade de processamento e potencial de cálculo desses computadores, seria possivel o controle de grandes plantas, inclusive com otimizações a nível global. De fato, é uma solução factível, com melhorias no desempenho. Porém surgiram alguns problemas novos. Por se basear em equipamento único, trazia de volta a questão da baixa confiabilidade. Soluções com redundância eram muito caras. Usavam-se novamente controladores de reserva. Em sistemas grandes e complexos, o número de variáveis é alto, da ordem de milhares

ou dezenas de milhares. Como decorrência, a entrada e saída torna-se complexa, necessitando de multiplexação de grande número de sinais. O software de apoio necessário (sistema operacional) é elaborado, com funções de escalonamento da UCP, gerenciamento de memória, tratamento de E/S, manutenção de base de dados e comunicação entre processos computacionais. Este software é de tamanho grande, gerando uma carga extra de processamento e comprometendo o tempo de resposta do controle.

Com o surgimento dos microcomputadores, a partir dos anos 70, foi possível implementar um método de controle que já vinha sendo considerado: o controle multinível e hierárquico (Kahne et al., 1980; Borsi e Pavlik, 1980). Neste método, o controle total é levado por um conjunto de elementos de computação hierarquizados, cada um com uma tarefa específica. No nível inferior, elementos de computação são encarregados de "interfacear" com partes do processo, fazendo aquisição de dados, transformando os dados coletados em formatos convenientes, transformando os dados calculados de controle e atuando no processo com esses dados transformados.

No nível imediatamente acima (intermediário), os elementos de computação são responsáveis pelo controle propriamente dito das partes do processo, executando algoritmos de controle a partir de dados oriundos do processo (nível inferior) e do nível superior. São funções de alta velocidade e críticas no tempo. Esses elementos de computação se reportam a outro computador situado no nível superior. As funções do computador deste nível são a coordenação do controle, incluindo o cálculo de valores de referência dos laços de controle, baseado em algoritmos de otimização de desempenho; a interação homem-máquina e homem-processo; a coordenação e suporte da comunicação entre elementos de computação do nível inferior; eventualmente, a supervisão e a otimização global. A medida que se sobe de nível, as funções vão se tornando menos críticas no tempo, porém vão ficando mais complexas.

A evolução dos sistemas digitais de controle, porém, caminha para uma distribuição ainda mais ampla das funções entre os elementos

componentes do sistema, de forma a relaxar a hierarquia existente entre o segundo e terceiro níveis. Estes sistemas, aqui chamados de Sistemas Digitais de Controle Distribuído - SDCD, têm a estrutura apresentada a seguir, juntamente com as funções desempenhadas em cada nível:

nível inferior - idêntico ao caso anterior;

nível intermediário - além das funções de controle em tempo real, são desempenhadas as funções de coordenação do controle, de interfaceamento homem-máquina e homem-processo e de provimento de mecanismos de comunicação com elementos do mesmo nível e do nível superior;

nível superior - desempenha as funções de supervisão global, gerenciamento de produção, otimização global, etc. Eventualmente provê o interfaceamento homem-máquina e homem-processo. E' um elemento opcional.

A utilização de arquiteturas de sistemas distribuídos, em particular as redes locais, é uma solução natural para o controle distribuído e para o hierárquico. A idéia é que os elementos de computação sejam os mais simples e específicos possível, tanto em termos de hardware como de software de apoio e de aplicação. Ademais, não existe nenhum elemento de computação que detenha, em caráter permanente, o controle do sistema como um todo.

2.4.2 Funções de um Sistema de Controle

Num sistema distribuído para controle, pode-se identificar uma série de funções que devem ser cumpridas a partir da preparação para operação de uma planta e durante sua vida. A seguir apresenta-se de maneira sucinta estas funções . Parte-se do princípio que um processo complexo é composto de processos parciais ou subprocessos (esta idéia será desenvolvida com mais detalhes adiante).

2.4.2.1 Aquisição de dados e atuação

A aquisição de dados, controle e atuação nos subprocessos constituem as funções básicas a serem executadas, semelhantes às funções dos controladores convencionais e computadores de controle digital direto. A aquisição consiste na leitura de sinais, analógicos ou digitais, que representam os valores das variáveis de processo - pressão, temperatura, nível, etc. Estes valores são resultantes de estímulos que o processo recebe, tanto de controle como de distúrbios. As variáveis são lidas e convertidas por algoritmos em formatos adequados para serem fornecidas às tarefas de controle. Os sinais analógicos são lidos periodicamente, dado que o processo é dinâmico. Os sinais digitais podem ser lidos periodicamente ou após a ocorrência de alguma condição interna ou externa. Por exemplo, uma interrupção externa para leitura de chaves indicadoras de condições de excessão.

Além do uso local da informação coletada, para cálculo de controle, esta pode ser usada em outros pontos do sistema, como, por exemplo, em monitoração centralizada. Para isso, a informação deve ser convenientemente enviada quando necessário, através do subsistema de interconexão. Desse modo, pode-se dizer que o sistema armazena informação de forma distribuída.

A atuação consiste na passagem dos valores calculados, após devidamente transformados, para os dispositivos atuadores no processo.

2.4.2.2 Controle

A função de cálculo do controle consiste na execução de algoritmos a partir de dados oriundos do processo (variáveis de processo); de dados calculados localmente por estimação; de dados

oriundos de componentes remotos do mesmo nivel e de niveis superiores, sendo tipicos os valores de referência para as variáveis de processo. Existe uma série de algoritmos de controle que podem ser usados isoladamente ou em conjunto, determinando o tipo de controle a ser aplicado. Cita-se como exemplos os controles por realimentação, cascata, "over-ride" e controle de sequência. A informação que determina quais algoritmos e como interconectá-los está localizada no elemento controlador em questão. Assim tem-se mais informação distribuida.

2.4.2.3 Configuração

Por configuração entende-se a atividade de escolha dos tipos de controle a ser aplicados a cada subprocesso e, consequentemente, dos algoritmos e da forma de interconectá-los. Os parâmetros dos algoritmos são definidos (ex.: periodicidade de leitura e atuação), bem como as adaptações a serem aplicadas nos dados de entrada e saida.

Esta atividade varia em complexidade e flexibilidade, conforme expõe Hofmann (1983), partindo da existência de blocos funcionais pré-programados para implementar algoritmos e funções de adaptação de entrada e saida. No caso mais simples e menos flexivel, a tarefa se constitui em escolher e interconectar apenas três blocos funcionais (entrada, algoritmo e saida). No outro extremo, a configuração consiste em escolher e conectar quaisquer números de blocos funcionais para implementar qualquer tipo de controle que se queira.

A configuração pode ser levada a efeito tanto local como remotamente. No primeiro caso atua-se diretamente no elemento fisico, para escolher, interconectar e eventualmente armazenar os blocos funcionais. No caso remoto, a atividade é feita de um outro ponto do sistema (uma estação) e a informação é passada ao elemento fisico através do subsistema de interconexão. Ainda, a

configuração se dá antes de colocar a planta em operação ou durante a operação da mesma (reconfiguração dinâmica).

2.4.2.4 Inicialização/atualização de valores de referência

Após a configuração do controle para cada sub-processo, antes da operação, é necessário fornecer os valores de referência das variáveis de processo ("set-points"). Esses valores podem variar ao longo da operação da planta, em função tanto de mudanças da política de produção como de resultados obtidos de cálculos para otimização da operação. Como consequência, faz-se necessário a atualização desses valores em tempo de operação.

2.4.2.5 Apresentação de informação

A obtenção de informações do processo sendo controlado e do sistema automatizado de controle é de importância fundamental para o bom desempenho e operação da planta. Neste sentido, um sistema de controle deve ser capaz de prover o acesso a dados situados em locais diversos, convenientemente preparados, sob demanda ou não. Tipicamente, no tocante ao processo, a informação requerida é organizada em uma hierarquia de formas de apresentação e é apresentada em diversos meios de saída. Em Weidlich e Prutz (1982) encontra-se uma lista algo semelhante à apresentada a seguir, porém mais detalhada. Assim tem-se:

- visão geral da planta, por áreas, de forma simbólica ou através de diagramas P/I;
- visão de grupos de controladores;
- visão de laços individuais de controle;
- anunciação de alarmes e/ou condições de excessão, associados a cada nível da hierarquia, sumarizados ou

agrupados.

Em muitos casos requer-se, além do valor medido corrente de uma variável de processo, uma curva de tendência da mesma (comportamento passado).

Do ponto de vista do sistema de controle, necessário se faz a indicação de falhas de componentes, da carga corrente do subsistema de interconexão, de erros de programa, da configuração corrente, entre outras.

A informação assim considerada se destina tanto ao uso por operadores como ao registro em dispositivos de armazenamento. No primeiro caso, os meios são, por exemplo, as telas de video, impressoras, traçadores gráficos e painéis de monitoração. No segundo caso, os meios são dispositivos de armazenamento secundário como discos e fitas magnéticas.

2.4.2.6 Manutenção de base de dados

O volume de informação tratada numa planta complexa é grande, e se compõe de dados que vão desde a configuração até o estado de variáveis. A disposição que se apresenta natural para estes dados é a distribuída, caracterizando uma base de dados distribuída. Decorrente deste fato, à semelhança dos bancos de dados distribuídos, surgem vantagens em relação à base de dados centralizada e também problemas novos. Entre estes estão os problemas de localização (tabelas, arquivos e diretórios); de formulação de consulta e/ou recuperação de dados; de concorrência (acesso simultâneo). Ademais, por estar em ambiente de tempo real, surgem problemas de atendimento critico no tempo.

2.4.2.7 Supervisão e otimização

Num esquema de controle distribuido ou hierárquico, o nível

superior é responsável pela supervisão do sistema como um todo integrado e por tarefas de otimização global da operação. Ao passo que no nivel inferior são executadas funções simples, rápidas e criticas no tempo, no nivel de supervisão e otimização são executadas funções mais complexas, possivelmente requerendo alto poder de cálculo e tratando alto volume de dados. Dentre essas funções pode-se destacar:

. otimização de desempenho;

. otimização de gastos com energia;

. otimização de consumo de matérias primas;

. gerenciamento de base de dados do processo sob controle;

. gerenciamento de base de dados de modelos de processos e seus controles, estratégias de produção, estoques, pessoal, etc.

A execução ou não dessas funções de forma automatizada depende tanto da capacidade do sistema de controle como da complexidade do processo. Pode-se dizer que este ainda é um campo de pesquisa.

2.4.3 Organização Funcional e Arquitetura

Procura-se nesta seção a identificação de tipos de componentes físicos existentes nos sistemas de controle distribuído, do ponto de vista funcional e enquadrados no esquema de controle hierárquico. Considera-se que os componentes são interligados por um subsistema físico de interconexão, a ser tratado posteriomente.

2.4.3.1 Controle, aquisição e atuação

As funções de controle, aquisição e atuação em um subprocesso são levadas a efeito por um computador, em geral um micro, referido como estação de controle local (ECL) ou controlador. O conjunto de estações de controle local constituem o subsistema de controle local (SCL). Uma ECL tem características de tempo real; sua capacidade se mede pelo número de malhas de controle tratadas bem como pelo número de entradas e saídas analógicas e digitais, além da velocidade de processamento.

Uma possível estrutura para a estação está apresentada na figura 2.8. Cada estação é composta de um conjunto de módulos funcionais, a seguir descritos.

módulos de entrada e saída de processo

São encarregados de receber e adaptar dados analógicos ou digitais do subprocesso, e de enviar ao mesmo os sinais de controle. Cada um se compõe basicamente de conversores A/D e D/A, de isoladores e possivelmente de um microprocessador e uma memória local para execução de algoritmos de transformação dos dados. Quando houver memória local, esta será conectada ao barramento da estação.

módulos de controle

São encarregados de executar os algoritmos de controle, a partir de dados originados nos módulos de entrada e saída de processo e de parâmetros locais. Os resultados são passados para os módulos de entrada e saída de processo. Possuem um ou mais microprocessadores. Os programas que implementam os algoritmos podem estar em memória local ou na memória de programa da estação.

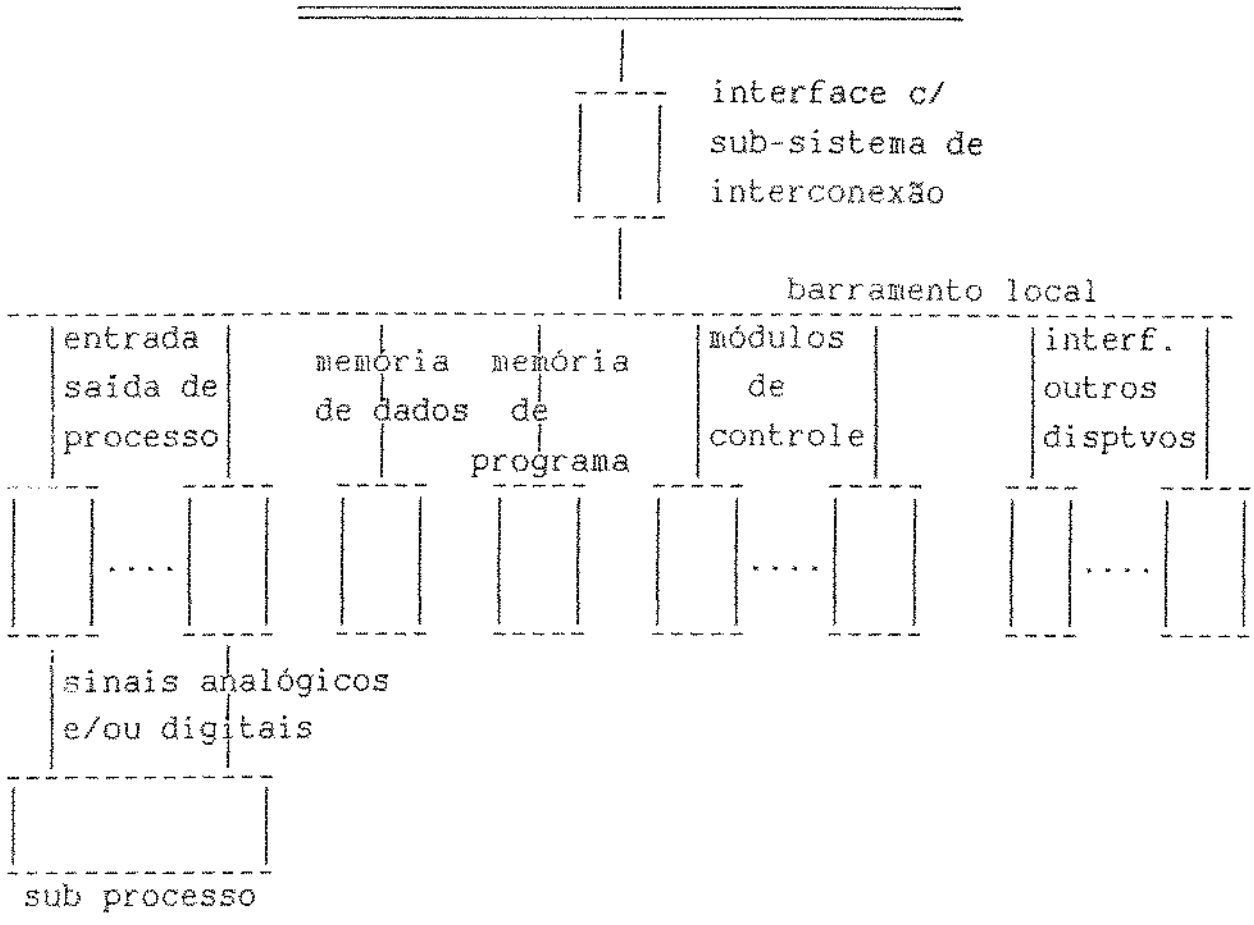


Figura 2.8 Estrutura geral de uma estação de controle local.

módulos de memória

Armazenam os programas (algoritmos, comunicação externa, sistema operacional local) e dados (parâmetros de algoritmos, configuração, valores de referência, valores de variáveis lidas, valores calculados de controle).

interfaces com outros dispositivos

Conectam a estação com dispositivos externos, de entrada (p.ex.: teclado), de saida (p.ex.: painel) e de armazenamento (p.ex.: unidade de disco flexivel).

interface com subsistema de interconexão

Encarrega-se de receber e processar mensagens oriundas do meio externo via subsistema de interconexão, bem como de enviar mensagens originadas na estação. Deve obedecer ao protocolo de comunicação do subsistema de interconexão. As mensagens recebidas são depositadas na memória de dados da estação. As mensagens enviadas são retiradas da memória.

A arquitetura da estação deve ser o flexível bastante para permitir o uso de redundâncias, tanto de módulos internos como do subsistema de interconexão.

De importância fundamental é a existência de um sistema operacional de tempo real, que tenha as seguintes caracteristicas, dependendo dos requisistos da aplicação:

- suporte de atividades ou tarefas em execução concorrente;
- utilização de um relógio de tempo real;
- facilidades de verificação de estado e escalonamento de dispositivos, para o caso de tratamento de falhas;
- tratamento de eventos, e consequente ativação de tarefas;
- suporte de mecanismos de sincronização para compartilhamento de recursos entre tarefas;
- facilidades de interfaceamento com instrumentos de entrada e saida de processo;
- facilidades de interfaceamento para comunicação com outras ECL's;
- facilidades de gerenciamento de memória principal e secundária.

2.4.3.2 Operação e monitoração

A presença do operador humano num sistema de controle é sempre necessária, tanto para monitorar o funcionamento da planta como para eventualmente operá-la. Identifica-se para tal fim uma ou mais estações de monitoração e operação (EMO) que, em conjunto, constituem o subsistema de monitoração e operação (SMO). As tarefas de monitoração e operação podem ser centralizadas em uma única estação, distribuidas por grupos de subprocessos ou mesmo localizadas em pontos especificos do sistema. As funções executadas por uma estação são, de uma maneira geral, a monitoração (apresentação de informação), o armazenamento de informação, a inicialização , cálculo e atualização de valores de referência, a configuração do controle e, também, o controle direto de subprocessos.

Os dispositivos usuais para entrada de dados são os teclados alfanuméricos, teclados de funções , cursores e "ligth-pen". Para saida usam-se telas de video, impressoras, registradores e traçadores gráficos, painéis, etc. Para armazenamento de dados, unidades de discos e fitas.

O porte e a complexidade da configuração de uma estação de monitoração e operação variam em função da utilização da mesma. Num esquema centralizado de monitoração e operação é requerido para a estação um maior número de dispositivos de entrada e saida, quando comparado com um esquema em que as estações estão localizadas fisicamente próximas dos subprocessos. Neste caso, é comum encontrar-se pequenas estações de "bolso", apenas com um teclado de funções e uma tela de video.

O sistema operacional deve prover facilidades variadas de entrada/saida e ser de tempo real, particularmente para o caso de operação via estação.

2.4.3.3 Supervisão e otimização geral

O esquema de controle distribuído abre a possibilidade de se lançar mão de um computador de elevado potencial de cálculo, com um sistema operacional de tempo real e de propósitos gerais. Este componente, opcional, caracteriza o subsistema de supervisão e otimização (SSO). Como dito anteriormente, as funções executadas são mais complexas e em geral menos críticas no tempo, com relação ao processo sendo controlado. Um dos objetivos é realizar funções de coordenação do controle dos vários subprocessos considerados como um todo integrado, com possíveis acoplamentos, podendo levar a otimizações de eficiência, de consumo e de custos associados. Um outro objetivo é a supervisão ou o gerenciamento da produção, onde se levam em conta aspectos como estoques, disponibilidade e encomendas de matérias primas, previsão de mercado comprador, disponibilidade de pessoal, utilização dos processos, etc. Pode manter um banco de dados com informação sobre, por exemplo, modelos de processos físicos e seu controle, estratégias de produção e estoques.

2.4.3.4 Comunicação e interconexão

O elemento de ligação das diversas estações caracteriza o sub sistema de comunicação (SCOM). E' responsável pela transmissão efetiva da informação trocada entre os diversos subsistemas. E' composto de um meio físico de interconexão, de interfaces com os outros subsistemas e de todo software de serviço necessário à efetivação da troca de informação. São aspectos relevantes a topologia de interconexão, o mecanismo de acesso ao meio e os protocolos de comunicação. Importante ressaltar que certas funções do SCOM , embora possam estar fisicamente implementadas em outros subsistemas (partes do software e interfaces), são consideradas como integrantes do mesmo. O SCOM pode ser visto como uma série de camadas funcionais e de serviço objetivando transportar informação da origem ao destino. As funções e serviços de cada camada são implementadas por software ou

hardware. A camada inferior tem a função de transmitir fisicamente a informação de um ponto a outro da rede e é implementada por meios fisicos de transmissão. A camada superior está diretamente relacionada com o usuário ou aplicação, implementada por software.

A este subsistema se aplicam as funções tratadas na seção de redes locais, tais como topologia, meios de transmissão, protocolos de controle de acesso ao meio e padronização.

2.4.4 O Modelo de Referência

No intuito de dar uma visão esquematizada e de conjunto, apresenta-se na figura 2.9 um modelo estrutural de referência de uma rede local para implementação de Sistemas Digitais de Controle Distribuido.

O modelo pode ser visto sob dois ângulos. No primeiro vê-se o sistema como um conjunto de subsistemas:

SCL - subsistema de controle local, composto de estações de controle local (ECL), as quais são funcionalmente divididas em módulos de controle, de entrada e saída de processo e de interface com o subsistema de comunicação;

SMO - subsistema de monitoração e operação;

SSO - subsistema de supervisão e otimização geral;

SCOM - subsistema de comunicação.

No segundo ângulo, vê-se o sistema organizado em níveis de hierarquia:

nível 1 - diretamente relacionado com o processo, com funções de aquisição de dados e atuação no processo;

nível 2 - implementa funções de cálculo e coordenação do controle, de interfaceamento homem-máquina e homem-processo;

nível 3 - implementa funções de supervisão global, gerenciamento de produção e otimização global.

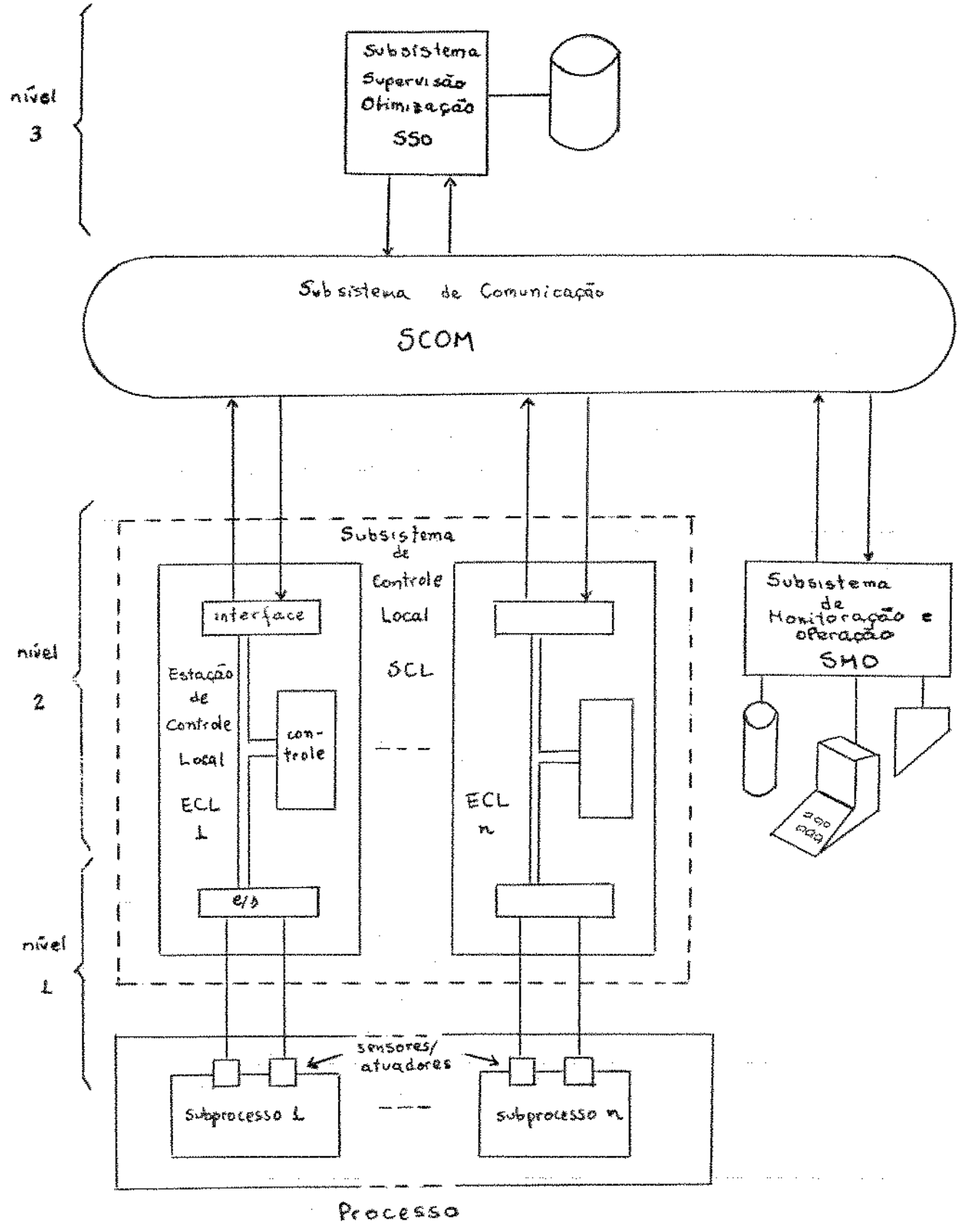


Figura 2.9 Modelo de referência de SDCD.

2.4.5 Requisitos e Atributos

Os sistemas distríbuidos para aplicações em controle de processos operam processos e plantas de alto custo. Por isso devem apresentar carcterísticas de alta confiabilidade e tolerância a falhas. Ademais, devem apresentar características de modularidade. O sistema deve sempre estar em funcionamento, a despeito de falhas de componentes e de sobrecarga no sistema de comunicação.

Um recurso necessário para se obter tolerância a falhas em cada uma das unidades de processamento e comunicação, é a redundância funcional ou dinâmica. Esta consiste em fazer determinadas unidades assumirem as funções de outras sob falhas. A redundância funcional é atingível quando a capacidade e desempenho das unidades forem apropriadamente super-dimensionadas para funcionamento em regime normal. A estratégia para se obter redundância funcional, conforme propõe Steusloff (1984), consiste em subdividir as funções de controle automático em três categorias de importância:

- funções indispensáveis (p.ex., segurança);
- funções parametrizáveis (p.ex., taxa de amostragem de laços de controle);
- funções temporariamente dispensáveis (p.ex., impressão de relatórios e atividades de desenvolvimento de programas).

Na ocorrência de falhas, a primeira providência é dispensar as funções da terceira categoria. Sendo insuficiente, as funções da segunda categoria deverão ter seus parâmetros alterados de forma a demandar menos capacidade e recursos dos sistemas de controle e comunicação, no sentido de liberá-los para um desempenho aceitável das funções da primeira categoria. O sistema passa a desempenhar suas funções de maneira degradada, à espera de correções e reparos. Este estado é designado de "gracefull degradation". Os sistemas que assim comportam apresentam o atributo de alta disponibilidade. Obviamente o sistema deve prover meios de detectar e localizar erros e falhas, uma tarefa importante para os projetistas das aplicações ou dos sistemas de controle.

2.4.6 Exemplos de Sistemas Industriais Existentes

Hoje já existe na indústria uma diversidade de sistemas de controle de processos industriais que contemplam aspectos dos SDCD's e aspectos das redes locais de comunicação. Em geral todos apresentam uma distribuição de dispositivos e funções de controle. Todavia, a comunicação entre os dispositivos nem sempre é feita por um meio comun de comunicação. Além disso, o controle da comunicação é, na maioria dos casos, centralizado. Cita-se sucintamente a seguir, exemplos de sistemas industriais existentes, juntamente com algumas de suas caracteristicas.

TELEPERM-M

Fabricado pela Siemens, possui um barramento geral ao qual podem ser acoplados até 32 barramentos locais. Nestes barramentos locais se ligam os dispositivos e/ou estações de controle e monitoração/operação. O controle da via é centralizado, mas o elemento que possui o controle (mestre) é dinamicamente trocado. O protocolo de transporte de dados é próprio, baseado em comandos e respostas. E' possivel a conexão com outros sistemas a barramento através de protocolo próprio ou do protocolo V.24. Tem aplicações gerais na automação de processos, sendo exemplo os processos químicos e energéticos.

PROVOX

Fabricado pela Fisher Controls, possui um barramento comum por área local, com controle centralizado de comunicação, sendo possivel acoplar até 30 dispositivos, configurando um subsistema local. Usa um protocolo para transmissão de dados semelhante ao HDLC. Os subsistemas locais podem ser interconectados via elemento ativo (configuração estrela). Serve para aplicações gerais de automação de processos.

CONTRONIC-P

Fabricado pela Hartmann & Braun, possui um barramento de controle centralizado, podendo conectar até 127 estações, das quais 32 ativas. O protocolo de transmissão de dados é do tipo PDV. Não apresenta o conceito de estação de supervisão e otimização. Possui boas caracteristicas de tolerância a falhas, permitindo alto grau de redundância em todos os niveis. À interligação com outros sistemas dá-se via protocolo HDLC.

Os SDCD's atuais da indústria são substituidores dos clássicos sistemas analógicos de controle, tendo sido concebidos gradativamente. Talvez por isso não apresentem ainda as caracteristicas gerais de SDCD baseados em redes locais de comunicação.

2.5 SISTEMAS DISTRIBUIDOS NA AUTOMAÇÃO DA MANUFATURA

As atividades da produção industrial de peças, produtos e máquinas vêm experimentando uma evolução tecnológica bastante acentuada nos aspectos de automação. O desenvolvimento vem a partir da automação rígida, de produção em massa com alta produtividade e pouca flexibilidade e caminha para os sistemas de automação programável, com o máximo de flexibilidade sem perder na produtividade. A automação programável (ou programada) se constitui de um conjunto de tecnologias, muitas das quais fortemente relacionadas com a informática. A idéia central em torno do termo programável é que a troca de tarefas de produção se dê por simples trocas de programas de computadores.

2.5.1 Funções e Evolução na Automação Programada

Esta secão trata de caracterizar de forma condensada a automacão da manufatura e de mostrar as tendências em termos tecnológicos da mesma. Boa parte desta seção se baseia em Mendes (1984).

O setor produtivo envolve três classes de funcões, quais sejam de projeto, de manufatura e de administracão e planejamento. Em todas elas a automação programada está ou pode estar presente.

A primeira classe refere-se ao projeto, onde se incluem as técnicas de CAD ("Computer Aided Design") e de CAE ("Computer Aided Engineering"). A segunda classe consiste de funções de manufatura, incluindo os equipamentos e sistemas para produção, como os robôs, as máquinas ferramentas a CNC ("Computerized Numeric Control"), os sistemas flexíveis de manufatura (FMS - "Flexible Manufacturing Systems") e os sistemas automáticos de manuseio, transporte, armazenamento, controle de qualidade, etc. Finalmente, na terceira classe, tem-se as funções próprias da administração e planejamento de produção, nas quais se incluem as técnicas de CAP ("Computer Aided Production") e CAPP ("Computer Aided Process Planning").

A tendência de evolução é a interligação dessas funções de forma integrada e harmônica, resultando nos sistemas integrados de manufatura (CIM - "Computer Integrated Manufacturing").

As aplicações da automação programável em uma fábrica vão desde a manufatura discreta (de peças simples a máquinas completas), passando pelos setores têxteis, de calçados, etc, até a manufatura de equipamentos complexos dos setores automobilístico e aeroespacial.

2.5.1.1 Fluxo produtivo na manufatura

A produção de um determinado produto envolve um conjunto variado de processos ou atividades. O fluxo produtivo se inicia com os resultados de pesquisa de mercado ou com a encomenda do produto. O passo seguinte é o planejamento das necessidades de materiais e da capacidade da manufatura. Estas especificações são utilizadas na próxima atividade, de projeto do produto. Nesta atividade busca-se a otimização de soluções, tendo em vista as restrições de materiais, processos de usinagem e tecnologias disponíveis. A próxima atividade consiste no planejamento do processamento (CAPP), incluindo as sequências de usinagem, escolha de máquinas, ferramentas, etc. Ao final tem-se, por exemplo, os programas de comando numérico para máquinas ferramentas.

A etapa seguinte é a de produção, constituída de grande número de processos. Entre estes estão o manuseio e transporte de materiais (por ex., com robôs), a usinagem (torneamento, aplainamento, etc), o acabamento, a estocagem e o despacho. De forma paralela ocorre o controle de qualidade do produto em confecção.

Estas atividades envolvem um conjunto de problemas básicos. Resumidamente, tem-se que tratar de um alto fluxo de informação, tem-se que buscar um alto grau de flexibilidade e tem-se que

chegar a elevados niveis de eficiência. A perspectiva de solução está em se lançar mão de computadores para todos os niveis da fábrica, através dos conceitos de automação programável.

2.5.1.2 Subsistemas empregados na automação programada

A seguir serão descritos sucintamente alguns dos subsistemas empregados na automação programada, e serão mostradas, dentro do possivel, as tendências de evolução.

CAD

Projeto assistido por computador, objetiva a concepção e projeto de um produto utilizando técnicas de computação gráfica. Um sistema de CAD se compõe de um hardware central, em geral um minicomputador, de uma diversidade de periféricos de E/S e de estações de trabalho (WS - "work stations"). O software é sofisticado, com sistema operacional próprio, uma base de dados gráficos e recursos de modelamento geométrico (2D, 2 1/2D, 3D).

Os sistemas de CAD evoluiram desde os sistemas de desenho a 2 dimensões até sistemas com metodologias modernas de modelamento sólido. Como tendência vê-se a melhoria dos algoritmos de representação de objetos, o aumento da inteligência e o provimento de facilidades de interligação com outros processos da manufatura. As estações de trabalho caminham no sentido de apresentarem alta resolução de imagem, interligação em rede, uso de linguagens naturais e de ténicas de inteligência artificial. O impacto desses futuros sistemas de CAD na automação da manufatura integrada deverão surgir na década de 90.

Sistemas de Manuseio - Robôs

Os sistemas de manuseio mais conhecidos são os robôs, que são manipuladores programáveis capazes de mover peças e ferramentas por trajetórias pré-estabelecidas. Um robô industrial se compõe de manipuladores, efetuadores, controladores (hardware e

software) e sensores. A programação das funções operacionais de um robô é atividade bem dominada. Aplicam-se em carga/descarga de peças e ferramentas em máquinas, no manuseio de materiais, na pintura, na soldagem, na usinagem, etc. A evolução caminha para o aumento de precisão de sensoriamento, diminuição de peso e rigidez e aumento da funcionalidade dos efetuadores. Deve ser aumentada a eficiência dos sensores, com relação a questões de processamento em tempo real. A programação evoluirá para o uso de linguagens específicas. Prevê-se o surgimento de robôs com vários braços, efetuadores flexíveis e programação via sistema de CAD. Num futuro mais distante surgirão robôs inteligentes, com visão 3D, funcionando em ambientes não estruturados.

Sistemas com Máquinas Ferrramentas a CNC

Comando numérico é uma técnica de automação de processos de fabricação executada por uma máquina ferramenta, onde todas as informações geométricas e tecnológicas necessárias para a fabricação são pré-codificadas e armazenadas em forma de programa, e passadas à máquina ferramenta de forma automática. O comando numérico dá flexibilidade para a fabricação, permite grande precisão, possibilita o controle remoto da fabricação, bem como possibilita o aumento de produtividade. No comando numérico existem funções de intertravamento e sequenciamento, interpolação e comandos por trajetórias pré-calculadas. A operação dá-se por atuação nos motores dos eixos das máquinas. Existem linguagens específicas para a programação (APT e seus dialetos).

O comando numérico evoluiu para o comando numérico computadorizado (CNC), à base de processadores digitais e memórias, com maior generalidade. O passo seguinte da evolução foi o comando numérico direto (DNC), com um computador controlando diretamente diversas máquinas. Os passos seguintes da evolução giram em torno do aperfeiçoamento das máquinas e também do comando (p.ex., comando de dezenas de eixos).

Espera-se também o provimento de interfaces com outros sistemas, o uso de máquinas ferramentas com controle adaptativo seguro, o

sensoriamento de ferramentas e a interligação de controladores com robôs. Finalmente, prevê-se a programação baseada em simulações 3D em sistemas de CAD.

Sistemas FMS

O passo seguinte na automação da manufatura em uma fábrica consiste nos sistemas flexíveis de manufatura (FMS). São sistemas que consistem no agrupamento de células de manufatura, integradas por sistemas automáticos de transporte e supervisionados globalmente por computadores em tempo real. Célula de manufatura é o agrupamento de diversas máquinas ferramentas a CNC com sistemas de manuseio automático. Nos sistemas FMS procura-se automatizar e controlar, por computador, instalações complexas, compostas pelo encadeamento de várias máquinas individuais. Objetiva-se a fabricação de peças com as mais diversas formas em uma sequência qualquer (flexibilidade) e com alta produtividade.

Um aspecto crítico nos sistemas FMS é a quantidade de software de tempo real a ser desenvolvido e integrado. A tendência é uma sofisticação cada vez maior, com supervisão à base de redes locais, diagnóstico automático de falhas e padronização de interfaces, por exemplo.

2.5.1.3 Manufatura integrada

A interligação desses diversos subsistemas para execução de funções em uma fábrica, desde a administrativa, passando por projeto e planejamento, até a fabricação, é uma possibilidade concreta. Um dos fatores decisivos é o desenvolvimento dos computadores e da tecnologia de comunicação entre computadores.

Uma fábrica moderna e automatizada hoje é vista como composta de subsistemas distintos e isolados - "ilhas de automação". Cada um tem suas funções específicas e têm como elemento integrador o homem. A fábrica do futuro, como parece ser a tendência, deverá ter como elemento integrador dos subsistemas o próprio

computador. Todavia, alguns problemas têm que ser primeiramente resolvidos. Entre eles, a utilização de protocolos de comunicação adequados e sua padronização e o desenvolvimento de sistemas de gerenciamento de banco de dados potentes e distribuídos.

2.5.2 Redes Locais na Automação da Manufatura

Um CIM pode ser visto sob dois ângulos. No primeiro vê-se os subsistemas que o constituem. Por um lado tem-se os subsistemas de CAD, CAP e CAM. Por outro lado tem-se os subsistemas de manufatura propriamente dita, entre os quais os robôs, as máquinas ferramentas a CNC e DNC e os FMS's. Estes dois últimos, principalmente, podem ser organizados segundo estruturas distribuídas. No segundo ângulo vê-se a interligação desses subsistemas. A idéia de rede local, considerando seu grau de generalidade e padronização da comunicação, parece natural para tornar viável esta interligação: uma diversidade de equipamentos e usuários a serem conectados a distâncias relativamente curtas, dentro de uma mesma organização. Aqui parece surgir uma dúvida quanto à especificidade da rede local: como compatibilizar os requisitos de tempo real das aplicações de robôs, máquinas ferramentas a DNC e sistemas FMS com os requisitos de comunicação que, pode-se dizer convencionais, das aplicações de CAD, CAM e banco de dados distribuídos ? A resposta é que os problemas de tempo real devem ser resolvidos internamente a cada subsistema. A comunicação via rede local integradora desses serviços dar-se-á para transmitir informação dos tipos arquivos, programas, consultas, resumos, etc. São serviços que, em geral, não demandam tratamento em tempo real, pelo menos no mesmo nível que nas aplicações anteriormente citadas.

Na figura 2.10 está representado um exemplo de um possível sistema CIM integrado através de rede local. Distinguem-se dois ambientes. O primeiro é o ambiente de escritório, onde são desenvolvidas as atividades relacionadas com projeto, planejamento, administração e mesmo com a elaboração de programas

de comando numérico. O segundo é o ambiente de fábrica, em torno dos equipamentos, relacionado com as atividades de produção. No ambiente de escritório ficam os usuários que trabalham, por exemplo, com CAD e CAP. Além das estações de trabalho respectivas e de equipamentos de serviços gerais como impressora e traçador gráfico, está mostrado um sistema de banco de dados distribuído ao longo de toda a rede.

No ambiente de fábrica estão representados alguns dispositivos controladores de máquinas ferramentas e robôs, bem como equipamentos de serviços gerais como impressora e terminais de video. Todos esses dispositivos se conectam à rede local. Cumpre notar a existência de um sistema flexível de manufatura (FMS) acoplado à rede local através de uma interface de redes ("gateway"). Este sistema tem seus componentes acoplados por um barramento local. A figura ilustra então o acoplamento de duas redes. Vê-se que é possível que toda problemática de tempo real envolvida no FMS seja resolvida internamente, conforme colocado no inicio desta seção.

2.5.3 Comentários

Aceito o fato que os requisitos de tempo real de determinadas aplicações possivelmente serão atendidos internamente em cada subsistema (comandos numéricos, controle de robôs, controladores lógicos programáveis, etc), pode-se levantar o perfil do tráfego de dados na rede local integradora. A informação que deve trafegar na rede consiste principalmente de textos, arquivos, programas e comandos/respostas de aplicações interativas do tipo consulta a banco de dados. Esse tráfego se caracteriza mais pela necessidade de alta vazão, em geral em surtos, do que pela necessidade de atrasos muito baixos de transmissão. Neste sentido a rede local integradora se assemelha às redes locais para aplicações de automação de escritórios.

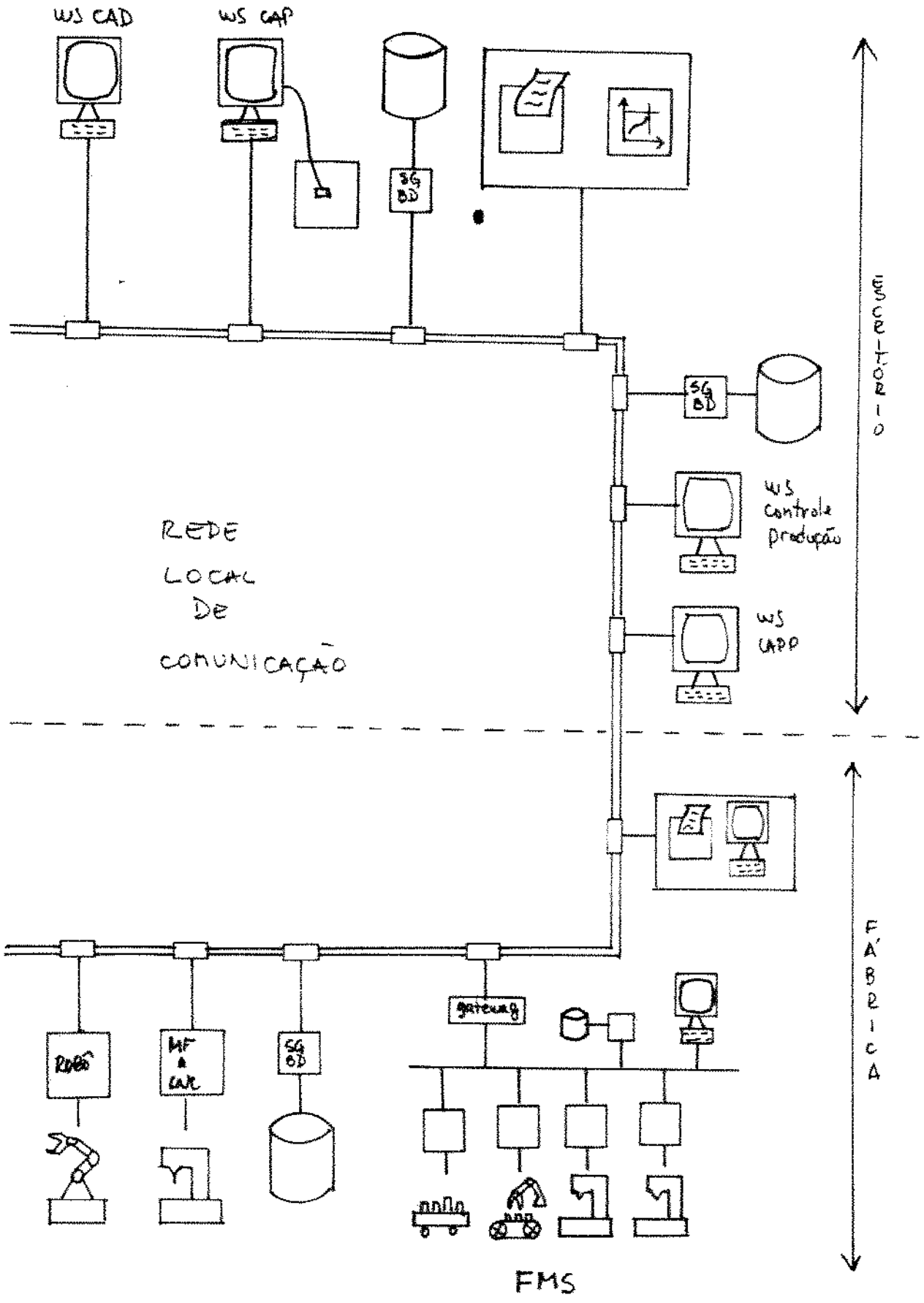


Figura 2.10 Redes locais na automação da manufatura integrada

O desenvolvimento de protocolos de níveis superiores para automação da manufatura integrada ainda está no seu início. Neste aspecto citam-se as facilidades para comunicação processo-processo, os protocolos de aplicações de banco de dados distribuídos, de transferência de arquivos e de outras aplicações interativas. O conjunto de protocolos específicos e necessários para a automação da manufatura estão se tornando conhecidos na literatura por MAP ("Manufacturing Automation Protocol").

2.6 CONCLUSÕES

Um objetivo deste capitulo foi a apresentação dos conceitos envolvidos em sistemas distribuídos e em redes locais de comunicação, tendo em vista a automação industrial. Entre os conceitos citam-se a arquitetura dos sistemas distribuídos, os protocolos de comunicação e as padronizações. O segundo objetivo consistiu na apresentação dos sistemas distribuídos para controle de processos industriais e, conforme visão do autor, a sua implementação com uso de redes locais. Essa parte terminou com a elaboração de um modelo de referência de sistemas distribuídos para controle baseado em redes locais.

O terceiro objetivo consistiu em apresentar o conceitos envolvidos na automação da manufatura e, de forma semelhante ao objetivo anterior, apresentar o relacionamento dos sistemas integrados de manufatura com as redes locais.

Pouco se falou a respeito de protocolos de níveis superiores para as aplicações tratadas no capitulo. A questão é que muito pouco existe sobre o tema, principalmente se forem considerados aspectos de padronização. No capítulo seguinte caminha-se neste sentido, onde é apresentada a especificação de mecanismos de suporte à comunicação processo-processo, para o caso particular de aplicações de sistemas distribuídos em controle de processos. Antes de se chegar à especificação é feito um levantamento das caracteristicas dessas aplicações a nível de comunicação.

3. COMUNICAÇAO E PROGRAMAÇAO EM SISTEMAS DIGITAIS DE CONTROLE DISTRIBUIDO

3.1 INTRODUÇAO

Este capitulo consiste na elaboração de proposições concretas no sentido de organizar e dar subsidios para a concepção da comunicação em sistemas distribuidos de automação, tendo em vista a implementação de camadas superiores da arquitetura de tais sistemas, a partir da camada de transporte da ISO, como será mostrado abaixo.

Primeiramente procura-se caracterizar a comunicação entre os componentes de tais sistemas, tomando como exemplo um SDCD. São estudados o tipo e a quantidade de informação trocada entre os componentes, bem como alguns requisitos de tempo, a titulo de ilustração.

Em seguida, é elaborada uma proposta de uma estrutura genérica de programas e de esquemas de comunicação para auxilio na concepção integrada de software de controle distribuido. A concepção é integrada no sentido de que todos os elementos envolvidos que concorrem para a consecução do controle, da operação e da supervisão de sistemas são partes de um mesmo conjunto. Isto porém não impede que eventualmente alguns deles existam como entidades isoladas e autônomas. A proposta, é importante ressaltar, não entra na área de modelagem de processos fisicos e seu controle, que constitui atividade executada à parte, numa etapa anterior do projeto.

O modelo proposto gera, por sua vez, um conjunto de requisitos que devem ser atendidos pela rede de computadores de suporte à implementação da aplicação de controle distribuido. Assim, parte-se para a especificação de um subsistema com o objetivo de viabilizar a comunicação a nivel de aplicação. Este subsistema especificado provê uma série de serviços à aplicação e torna transparente à mesma a existência de uma rede fisica de

comunicação. A especificação acontece em duas etapas: uma mais geral e informal e outra mais detalhada, próxima da implementação.

Antes porém, convém definir o escopo dos temas tratados neste capitulo, à luz da arquitetura em camadas dos sistemas distribuidos. Inicialmente, com relação às camadas inferiores, o trabalho está inserido acima da camada de enlace lógico definida pelo Comitê 802 do IEEE, bem como da camada de rede definida pela ISO. Com respeito ao usuário ou programador de aplicações especificas, na camada de aplicação da ISO, o trabalho se relaciona porque é definida uma estrutura de programas da aplicação.

A camada de apresentação, que vem hierarquicamente logo abaixo da camada de aplicação, objetiva a execução de funções frequentemente requisitadas, no sentido de se obter representações comuns para textos, caracteres, dados , etc. Comporta também a compressão de dados e codificações especiais para manutenção da segurança ao acesso da informação transmitida. Embora possa ser importante em aplicações de sistemas de automação industrial, principalmente quando há heterogeneidade de hospedeiros, esta camada não será especificamente tratada neste capítulo.

A camada de sessão visa o estabelecimento dinâmico de facilidades para que duas entidades do nível superior possam comunicar entre si, o que caracteriza uma sessão. Deve prover a sincronização do diálogo e implementar o endereçamento dos comunicantes. Estes aspectos são tratados neste capitulo.

A camada de transporte deve tornar transparente à aplicação a existência de uma sub-rede de comunicação e prover serviços de transporte de informação fim-a-fim. A camada aqui especificada isola a aplicação da sub-rede, através do mapeamento nome-de-processo/endereço-de-nodo. Assim, para os requisitos de comunicação das aplicações consideradas, fica opcional a inclusão de uma camada de transporte. Como dito no início desta introdução, os requisitos que a presente camada deve atender são originados das propostas do modelo de programas e de esquemas de

comunicação.

As atividades de padronização dos serviços das camadas superiores estão ainda em fase de busca de soluções. Primeiro, não está claro ainda onde classificar, se na camada de aplicação ou na de apresentação, os serviços do tipo "terminais virtuais" e "transferência de arquivos". Segundo, também não está claro a classificação de serviços como comunicação e sincronização entre processos (IPC), se na camada de transporte ou na de sessão. Todavia, neste particular, neste trabalho optou-se por colocar estes serviços de comunicação e sincronização na camada de transporte.

Um outro aspecto que merece destaque é o da forma e local de implementação dos algoritmos que executam as funções tratadas neste capitulo. Três opções existem:

a primeira delas, mais simples e direta, consiste na elaboração de rotinas para executar as funções. Estas rotinas são ligadas ao programa do usuário, podendo ou não acessar as facilidades básicas de entrada/saída do sistema operacional local;

a segunda consiste na incorporação das rotinas ao sistema operacional local, acrescentando características de distribuição a este. Esta opção evolui para a concepção de sistemas operacionais distribuídos.

a terceira opção vai no sentido de incorporar, a linguagens de alto nível, características de comunicação entre processos remotamente situados.

As duas últimas opções constituem ainda campo vasto de pesquisa. A opção pela incorporação das funções a sistemas operacionais parece mais adequada. Primeiro, um sistema operacional serve para esconder detalhes do usuário (compiladores e programas de aplicação) e dar a este facilidades de uso e acesso a recursos do sistema computacional. Considerando o sistema de comunicação como

um recurso, a conclusão é natural. Segundo, o compilador de uma linguagem de programação que pretenda ter um certo grau de generalidade, não deve se ater a aspectos particulares de aplicações. Todavia a linguagem deve ter definidas as operações primitivas para que a comunicação se efetue.

Enfim, neste capítulo pretende-se dar subsídios metodológicos para a concepção de aplicações de automação em sistemas distribuídos, tomando-se como pano de fundo os Sistemas Digitais de Controle Distribuído para o controle de processos industriais. Além disso, pretende-se contribuir em aspectos de comunicação, a nível de especificação, projeto e implementação de protocolos para suporte à comunicação entre processos computacionais.

3.2 CARACTERIZAÇÃO DA COMUNICAÇÃO A NIVEL DE APLICAÇÃO

Como dito, optou-se por abordar o assunto a partir de SDCD em controle de processos industriais. Os diversos componentes de um SDCD trocam informação visando fins distintos. A quantidade de informação trocada de cada vez e os requisitos de tempo de resposta variam conforme o caso. À seguir procura-se caracterizar a comunicação entre os componentes de um SDCD.

3.2.1 Comunicação SMO_ECL

Dois tipos de informação podem ser identificados na comunicação SMO_ECL, que são de monitoração e de operação.

A informação de monitoração serve para indicar o estado corrente da operação do sistema de controle. Consiste basicamente dos seguintes tipos de dados:

- de indicação do estado das variáveis dos subprocessos sob controle;
- de indicação do estado das entradas digitais;

. de indicação de situações de excessão e/ou alarmes;
. de indicação do estado dos componentes do sistema de controle (operação normal, falha, degradação).

Para caracterizar a informação de monitoração quanto ao tamanho das mensagens transmitidas é preciso algumas considerações iniciais. Da literatura referente a SDCD's existentes, entre os quais os sistemas MAX1 (Leeds & Northrup, 1982), DS100 (ASEA, 1982) e Network 90 (Bayley Control, 1982), pode-se dizer que os números de entradas e saídas manipulados pelas ECL's ficam em torno dos seguintes valores:

. entrada analógica = até 32
. saída analógica = 8
. entrada digital = 256
. saída digital = 256

Em função desses valores pode-se estimar os tamanhos das mensagens. Considerando uma dezena de "bytes" para prefixo e sufixo das mensagens, que é um valor razoável para redes locais, quatro "bytes" para representação de cada valor analógico e um "bit" para digital, tem-se os seguintes tamanhos de mensagens ("bytes"):

. =< 138 p/ indicação de estado de variáveis
. =< 42 p/ indicação de estado de entradas digitais
. =< 138 p/ indicação de situações de excessão e/ou alarmes

Estes valores são para os piores casos, onde toda informação é passada de uma só vez. Mensagens de indicação do estado dos sistemas de controle e de comunicação tendem a ser mais reduzidas. Estas estimativas estão de acordo com Jensen (78), que estipula mensagens com tamanhos variando de 20 a 200 "bytes".

O sentido do fluxo de mensagens pode ser bidirecional ou unidirecional de ECL para SMO. O primeiro caso ocorre quando o SMO solicita da ECL os dados para monitoração, de maneira não determinística através de mensagens curtas. Após receber a

solicitação, a ECL responde enviando uma mensagem com os dados pedidos. O segundo caso, unidirecional no sentido ECL - SMO, ocorre quando os dados necessários são enviados periodicamente.

Os atrasos permitidos para que mensagens originadas nas ECL's cheguem a uma estação de monitoração dependem de alguns fatores. Quando ações, decorrentes da constatação de alteração de estado de um processo, forem tomadas por operador humano, certamente os requisitos de atraso de transmissão serão mais frouxos, compatíveis com o tempo de resposta deste operador. Se, por outro lado, estas ações forem tomadas automaticamente pela máquina, que obviamente responde mais rapidamente, os requisitos de atraso podem ser mais críticos. Tudo porém depende das constantes de tempo do processo em questão, como será visto mais adiante. Outro fator determinante é a capacidade de atualização da estação de monitoração, a qual recebe e processa mensagens de vários pontos. Um exemplo deste caso é apresentado por Schoeffler (1984): o atraso permitido ao sistema de comunicação pode variar de 0,1 a 1,0 segundos.

A informação de operação, que flui no sentido SMO_ECL, é necessária para manter o sistema em funcionamento normal ou sob anormalidades. A operação, enquanto comandada pelo SMO, pode ser manual ou automática. Em ambos os casos as ECL's recebem os seguintes tipos de dados:

- de manipulação de parâmetros de controle ou valores de referência;
- de manipulação de saídas digitais;
- de controle de variáveis de processo.

Os dois últimos tipos são transmitidos quando as ECL's forem incapazes de executar as funções de controle, nas inicializações e reinicializações ou em casos de ações para manter a segurança. No caso de transmissão de parâmetros, a comunicação dá-se em geral sem solicitação específica, ocorrendo de maneira não determinística. No caso de manipulação de saídas digitais e controle de variáveis de processo, ela pode ser periódica ou não.

Quanto ao tamanho das mensagens, aplicam-se as mesmas considerações do caso de monitoração.

A urgência da comunicação, ou o tempo requerido para que uma mensagem vá da origem ao destino, depende das constantes de tempo do processo, que nos casos usuais são da ordem de segundos ou mais. Isto se deve ao período de amostragem nas ECL's, que varia para cada malha de controle entre 0,2 a 1 segundo (Tozzi et alii, 1983). Não se descarta, entretanto, a possibilidade de controle de processos mais rápidos, o que já está acontecendo com SDCD's de tecnologias mais avançadas. Um tipo de aplicação onde se encontram processos rápidos é a de siderurgia, em particular na automação de laminadores. Aplicações que usualmente necessitam de Controladores Lógicos Programáveis demandam também um controle a respostas rápidas.

3.2.2 Comunicação ECL_ECL

A comunicação entre ECL's é ainda pouco explorada, sendo prática comum a operação das mesmas de forma desacoplada. Um dos objetivos deste trabalho é estudar os mecanismos de comunicação que facilitem a operação integrada das ECL's. Dois tipos de informação podem ser identificados nesta comunicação, quais sejam, de estado de variáveis e de sincronização.

A troca de informação de estado de variáveis entre ECL's ocorre quando suas operações são executadas de maneira integrada, na qual uma ECL envia a outra o estado de certas variáveis lidas localmente, recebidas do SMD ou geradas internamente pelo algoritmo de controle local.

Seja, por exemplo, o caso de controle de laminadores contínuos (Copeliovitch et alii, 1983). O processo consiste de vários subprocessos em sequência, nos quais o produto obtido em um subprocesso serve como produto primário (de entrada) para outro.

Uma sequência tipica é composta pelos subprocessos de aquecimento, de redução de seção e de confecção de bobinas. Havendo uma ECL para o controle de cada subprocesso, as seguintes trocas de informação são desejáveis. A ECL controladora do aquecimento fornece para a ECL controladora da redução informações tais como temperatura, velocidade e dimensões do produto que está sendo enviado. A ECL de redução, de posse destes dados recebidos e de dados obtidos localmente (especificação do produto e estado do subprocesso), tem condição de efetuar o controle (cálculo de velocidades, torque de motores, posicionamento de rolos, etc) e a atuação. Isto pode eliminar a necessidade de se fazer identificação e estimação de parâmetros, que muitas vezes são atividades onerosas, tendo em vista o tempo de processamento envolvido (Nogueira & Lages, 1983). Por outro lado, a ECL de redução pode mandar informação de volta à ECL de aquecimento, com o objetivo de que esta forneça um produto de maneira mais adequada ao melhor funcionamento da ECL de redução.

O tamanho das mensagens pode ser considerado pequeno. Seja o caso de uma ECL onde são controlados 8 pontos, através de 8 laços de controle, e cujos valores medidos são passados a outra ECL. Haveria então 32 "bytes" de dados a serem transmitidos, considerando que cada variável ocupe 4 "bytes". A troca de mensagens pode se dar a intervalos regulares, sem um diálogo, ou através de solicitações explicitas. O sentido do fluxo de mensagens pode ser uni ou bidirecional.

A troca de informação de sincronização ocorre quando as operações integradas das ECL's levam em consideração o tempo ou hora real. Seja o caso em que determinada ação deva ser tomada em conjunto por várias ECL's, num determinado instante de tempo futuro. Uma das ECL's, a que tem a informação do momento de atuação, envia às outras uma mensagem que especifica tal momento. Cada uma das ECL's restantes, após receber a mensagem e após um processamento associado, informa à emitente que está apta a atuar. A partir daí passa a esperar uma notificação da emitente para efetivar a atuação. Deve-se observar que a hora real em sistemas distribuídos carece de precisão. Pelo fato de cada elemento ter

seu próprio relógio, há sempre o risco de descompasso entre as medidas do tempo. Se o sistema tiver um relógio central e enviar periodicamente a medida do tempo, além de perder as caracteristicas de distributividade ainda ocorre o descompasso, pois os tempos envolvidos na transmissão são de natureza não determinística. Entretanto pode-se considerar faixas de valores que atendam aos requisitos do controle.

As mensagens de sincronização tendem a ser curtas, pelo fato de que contêm apenas dados indicativos de tempo, de ações a serem tomadas e respostas breves. A urgência na transmissão é função dos intervalos de tempo considerados, porém há sempre um limite, que pode ser critico, a ser respeitado para que a sincronização realmente ocorra. O sentido do fluxo de mensagens é, via de regra, bidirecional.

Embora os tipos de informação vistos sejam de naturezas distintas, há a possibilidade de que uma mensagem contenha os dois casos. Todavia isto não implica que as mensagens se tornem necessariamente grandes.

3.2.3 Outras Combinações

O restante das combinações de componentes que se comunicam consiste das comunicações SSO_ECL e SSO_SMO. A primeira é de ocorrência rara em regime de operação normal. Pode se dar em casos de configuração inicial do sistema de controle, onde os códigos dos algoritmos são passados do SSO para as ECL's e então carregados em memória principal, volátil ou não, para execução das funções de controle. Também ocorre nos casos de mudança de estratégia de controle com consequente mudança de algoritmos, como resultado de otimizações e de decisões de supervisão e gerenciamento. Em casos de anormalidades, quando o SMO falha, o SSO pode assumir as funções daquele e passar a comunicar com as ECL's da forma vista anteriormente. Deve-se observar que um sistema de controle distribuido que permite tais facilidades tem

um elevado grau de automatização, e é, em sua forma plena, um objetivo ainda a ser alcançado, constituindo um campo de pesquisa.

A segunda combinação restante, entre SSO e SMO, tem por objetivo principal a troca de informação para adequar o sistema de controle aos requisitos de eficiência, gerência e produção. Por exemplo, o SMO envia periodicamente ao SSO resultados da operação de cada componente do sistema de controle (ECL). De posse dos dados o SSO submete-os aos seus algoritmos de otimização, os quais consideram a operação integrada dos componentes. Eventualmente envia ao SMO informação para atualização do controle de um ou mais componentes, incluindo aí novos valores de referência, se for o caso. Uma função típica e adequada ao SSO é o recálculo de valores de referência. A diversidade de tipos de informação a ser trocada nesta combinação é grande, função do grau de automatização e da capacidade de cada SDCD.

Finalizando, pode-se dizer que, em regime de operação normal, as necessidades de comunicação podem ser atendidas com o que há de hardware e software disponível no mercado (redes físicas e seus protocolos). Porém em operação anormal ou em casos de falhas, onde um componente substitui as funções de outro, aumentando o fluxo de informação na rede, as exigências em termos de facilidades de comunicação, tempo de resposta e vazão de comunicação são maiores. Ademais, quanto maior o grau de automatização, maior o nível de comunicação e maiores as mesmas exigências.

Ainda, uma análise idêntica a esta pode ser feita para contemplar cada tipo de sistema distribuído para aplicações específicas. Cita-se como exemplo, os sistemas CAD/CAM. Os resultados de cada análise podem e com certeza serão diferentes - quanto ao tipo de informação transportada, quanto ao tamanho típico de mensagens, bem como quanto aos requisitos de tempo. As análises não foram feitas por exiguidade de tempo e espaço. Todavia é um trabalho que deve ser levado à frente para dar prosseguimento a estas

pesquisas.

3.3 UM MODELO DE PROGRAMAS PARALELOS PARA A APLICAÇÃO

3.3.1 Estrutura dos Sistemas de Controle

Uma das características dos modernos sistemas de controle distribuído é que oferecem a possibilidade, particularmente importante, de construção de qualquer sistema de controle complexo em uma maneira que é claramente organizada de acordo com estágios de hierarquia (Borsi & Pavlik, 1980). No início deste capítulo foi feita referência à modelagem de processos físicos e suas funções de controle. Aqui considera-se exclusivamente o modelo do sistema de software para a programação do controle em SDCD's. O modelo é apresentado como uma proposta para auxílio na concepção de tais sistemas, de forma integrada, nos aspectos que envolvam comunicação.

Um sistema de controle cumpre melhor seu papel na medida que sua estrutura reflita a estrutura do processo físico a ser controlado. Esta é uma regra geral aplicável à solução de problemas por computador, onde se busca uma solução cuja estrutura reflita a estrutura do problema. Um sistema a ser automaticamente controlado consiste de um processo físico distribuído ao longo de uma área, constituído de processos físicos parciais ou subprocessos. Para se chegar a uma estrutura da solução do controle de determinado processo, é útil adotar certos critérios de desenvolvimento e concepção. Um critério que parece adequado é apresentado por Steusloff (1979), denominado "princípio de dualidade" que, com relação ao processo físico e ao seu sistema de controle automático, estabelece o seguinte:

> "um processo físico automaticamente controlado consiste de partições distribuídas, com operações paralelas e acopladas (subprocessos). O sistema dual de controle consiste, pelo menos, do mesmo número de processos computacionais correspondentes, paralelos e igualmente acoplados".

Com esse critério em mente quando da concepção da solução, a estrutura do sistema de controle tenderá a refletir a estrutura do problema. Considerem-se, por exemplo, dois subprocessos num esquema sequencial, como no exemplo do laminador, nos quais o comportamento de um, concretizado pelo produto físico resultante, influi no comportamento do outro através da demanda surgida em função deste produto. Estão pois acoplados. O sistema de controle terá pelo menos 2 processos computacionais controladores correspondentes, acoplados de modo que um forneça valores de suas variáveis de estado ao outro.

Convém neste momento esclarecer um ponto que pode gerar dúvidas. Muito embora a estrutura do sistema de controle possa ser organizada hierarquicamente, a estrutura do sistema de comunicação não o é necessariamente. Não existe nenhum vínculo estabelecido "a priori" que condicione uma estrutura à outra. Assim o sistema de controle pode ser hierarquizado e o sistema de comunicação ser totalmente distribuído.

3.3.2 Estrutura do Modelo

Todos os componentes do software de um sistema que tratam especificamente do controle de um processo, incluindo monitoração, operação, otimização e supervisão, constituem o que é conhecido genericamente como programas de aplicação ou simplesmente aplicação. Com isto pretende-se dizer, por exemplo, que a programação que viabiliza a comunicação entre os vários componentes de um sistema distribuído forma uma parte distinta e independente de processos físicos e seus controles.

A proposta que será aqui apresentada refere-se aos programas da aplicação. Como visto anteriormente, a estrutura do sistema de controle deve refletir a estrutura do processo a ser controlado. Um processo industrial complexo consiste de partições distribuídas operando paralela e acopladamente, segundo o princípio de dualidade. O modelo de programas paralelos considera a existência de partições com operações também paralelas e

acopladas. Ao conjunto dos programas (aplicativos) que implementam uma aplicação especifica e que são alocados e executados nos diversos componentes do sistema distribuido (elementos de computação), dá-se aqui o nome de programa paralelo. Um programa paralelo é constituido de módulos. Os módulos quando em execução ou ativos são constituidos de processos. A comunicação entre módulos é feita através de mensagens.

Um módulo é um programa sequencial ou concorrente, que corresponde a um conjunto de tarefas afins como, por exemplo, aquisição, regulação e atuação. Neste caso o módulo está associado a um subprocesso fisico, o qual controla. Um módulo pode corresponder às tarefas de monitoração e operação, bem como às de otimização e supervisão. Não existe necessariamente uma relação biunivoca entre módulos e tarefas. Por exemplo, se o SMO for composto de duas estações, pode haver, para cada uma, um módulo para executar suas tarefas. Por outro lado, ainda exemplificando, as tarefas de operação e monitoração podem ser executadas por mais de um módulo.

Cada módulo tem seu próprio espaço lógico, ou seja, módulos distintos ocupam espaços de endereçamento independentes. O conjunto de nomes (variáveis, rótulos) referenciados por um módulo não guardam nenhuma relação com o conjunto de nomes referenciados por outro módulo qualquer. Este conceito é usual em sistemas com multiprogramação (Guimarães, 1980). Pode-se referenciar a módulos também como programas. A organização dos módulos na estrutura do programa paralelo pode ser hierárquica. Por exemplo, no caso de um processo complexo, módulos controladores de um conjunto de subprocessos podem estar subordinados ao módulo de uma determinada estação de monitoração e operação.

Processos são componentes sequenciais de módulos em execução. Ao ser posto em execução, um módulo evoca um ou mais processos. Os processos estão associados a recursos, como por exemplo: interfaces de entrada e saida de processo, dispositivos de

entrada e saída convencionais, processadores, sistema de comunicação, etc. Assim tem-se processos de entrada e saída, processos controladores, processos de comunicação e outros.

Mensagens são estruturas de dados destinadas à transferência de informação entre módulos. Uma mensagem é composta de dois tipos de informação. O primeiro serve para efetivação da transferência física da mensagem e contém dados de endereçamento, controle e categorização da mensagem. O segundo tipo contém os dados de interesse dos módulos comunicantes.

A comunicação entre módulos dá-se efetivamente entre processos especiais dos mesmos. Desse modo, de agora em diante, a referência à comunicação entre processos significa a comunicação entre módulos do programa paralelo. A comunicação interna a módulos não é de interesse agora, pois o aspecto em questão é a troca de informação entre os módulos que implementam um sistema de controle.

A configuração do programa paralelo, do ponto de vista do número de módulos é estática, ou seja, os módulos são todos conhecidos a partir da concepção do programa paralelo. Esta idéia é também apresentada por Brinch Hansen (1978) ao caracterizar programas de tempo real. No caso de SDCD a afirmativa procede pelo fato de que o programa é concebido a partir do modelo de processo, onde são identificados os subprocessos e as funções de controle a aplicar. Um problema surge: como compatibilizar a idéia de estaticismo de módulos frente às características de reconfiguração dinâmica que um sistema de controle deve apresentar? Quando ocorre uma falha em uma ECL de modo que ela não possa mais executar suas funções, outra parte do sistema deve assumi-las para garantir a continuidade da operação (redundância funcional). Nesse caso, deve ser tarefa do projetista do programa a previsão de módulos redundantes e mecanismos de passagem de controle, de modo a se obter um sistema com um grau de confiabilidade satisfatório. Um outro aspecto diz respeito a expansões do sistema em tempo de execução. Até onde for possível prever, a concepção de módulos para uso futuro é tarefa do

projetista. No caso de expansões que descaracterizem o sistema original, propõe-se a concepção de novo programa paralelo para substituição do corrente. Deve-se notar que grande parte dos problemas de recongifuração são de natureza de comunicação, nos aspectos de endereçamento e localização de módulos. Assim, muito da capacidade do sistema é função das características do suporte de comunicação.

O número de módulos de um programa paralelo é dependente da estrutura do processo sob controle. Em geral corresponde, em ordem de grandeza, à quantidade de subprocessos nos quais o mesmo é particionado. Existem processos com a ordem de centenas de subprocessos, como pode-se ver na literatura de fabricantes citada anteriormente e no documento elaborado pela SEI (1982). No modelo, as atividades de criação/deleção de módulos inexistem ou são raras.

O ambiente de execução do programa paralelo, com relação aos elementos de computação, é conhecido. Sabe-se de antemão as características de hardware e software auxiliares disponíveis (desempenho, recursos de armazenamento e de entrada/saída, etc). Este conhecimento do ambiente não implica necessariamente que no projeto do programa a alocação dos módulos seja feita concomitantemente com a concepção de cada um.

Todos os módulos operam paralelamemte, com início quasi-simultâneo ou segundo um sequenciamento estabelecido. A simultaneidade absoluta é um conceito que em sistemas distribuídos não existe na prática devido aos problemas de determinação do tempo, anteriormente referenciados. Porém, pode-se esperar que, dentro de uma certa tolerância, todos os módulos tenham sua execução iniciada. Mesmo os módulos que efetivamente não estejam gerando resultados, como os redundantes em estado suspenso, são inicializados e passam a esperar a ocorrência de alguma condição para prosseguimento de suas operações. Desse modo estão em estado de execução. O funcionamento de um módulo é, em geral, contínuo, sem término e de caráter cíclico, como o são os subprocessos. No caso de se considerar término, aplicam-se os mesmos conceitos das inicializações.

Os módulos do programa paralelo em execução tratam de eventos de natureza não determinística no tempo. Como consequência, as mensagens decorrentes do acontecimento de tais eventos podem chegar para um módulo a qualquer instante. Paralelamente existem nos módulos funções de execução periódica (ciclica), o que pode gerar troca de mensagens a intervalos de tempo constantes. Um exemplo dá-se na comunicação SMO_ECL no caso de envio de informação do subprocesso amostrado periodicamente. Estes aspectos de comunicação, incluindo sincronização, serão estudados com mais detalhes na seção seguinte.

3.3.3 Mecanismos de Comunicação

Num ambiente de controle distribuído os módulos executam ações concorrentes ou paralelas a fim de atingir o objetivo de um controle distribuído e integrado. Para que essas ações sejam cooperativas é necessário que haja comunicação entre processos de um módulo com processos de outros. Numa seção anterior discutiram-se essas necessidades em suas várias formas, e foram citados aspectos de sincronização.

Existem basicamente dois mecanismos para se efetuar a comunicação entre processos dos módulos de programas paralelos. O primeiro é o mecanismo de comunicação via memória comum compartilhada que consiste em fazer a comunicação (e sincronização) através de operações sobre dados comuns (variáveis globais) aos processos que assim se comunicam. Esse mecanismo é bem conhecido e largamente usado em sistemas com multiprogramação e mesmo com multiprocessamento. Os conceitos são bem dominados, entre os quais citam-se os de semáforos, regiões críticas e monitores. Estes conceitos estão implementados em algumas linguagens de programação como o Pascal Concorrente que tem embutido o conceito de monitor (Brinch Hansen,1975).

Este mecanismo requer para sua implementação o uso de um espaço

comum de endereçamento (espaço lógico ou de nomes) para os processos. A sincronização entre processos, que se dá pela troca de informação de controle, é de baixo custo desde que se tenha uma memória comum (Jones & Schwarz, 1980). O acesso aos dados compartilhados é, em geral, rápido. Enfim, este mecanismo vem sendo utilizado há bastante tempo em sistemas com centralização de memória, as idéias estão sedimentadas e os problemas resolvidos, como pode-se ver em Brinch Hansen (1979). Para sua utilização na comunicação em sistemas distribuídos surgem alguns problemas:

A exigência de um espaço de endereçamento comum para os processos resulta numa centralização do sistema, o que, para aplicações em sistemas distribuídos, não é desejável;

Para que processos se comuniquem segundo este mecanismo há necessidade de sincronização. Para que a sincronização ocorra deve haver troca de informação de controle entre o elemento que gerencia a comunicação e os processos. Como esta troca de informação dá-se via de regra através do subsistema de comunicação, este mecanismo tende a impor um sobreprocessamento à sincronização. Isto acarreta, por vezes, bloqueios longos a um número grande de processos, o que pode ter consequências não desejáveis em aplicações de tempo real;

A sincronização fica altamente vinculada a problemas de provimento de uma comunicação confiável sobre subsistemas de comunicação que podem não ser totalmente confiáveis. Tal problemática não surge nos sistemas centralizados porque o acesso à memória comum é de altíssima confiabilidade. Por exemplo, a utilização de semáforos é passível de erros devidos a falhas de transmissão não detectadas. Sendo o semáforo um contador, seu conteúdo pode se tornar inválido para sempre, se uma instrução que resulte na alteração do mesmo for perdida na transmissão.

Enfim, a adoção deste mecanismo, que implica em técnicas de controle centralizadas, descaracterizaria os sistemas

distribuídos tal como foi colocado no início do capítulo anterior.

O segundo mecanismo é o de transferência explícita de mensagens (ou de mensagens passantes), que consiste em fazer passar a informação entre processos em forma de mensagens, sem a restrição que estas mensagens sejam estruturas de dados comuns aos processos. Quando um processo envia uma mensagem a outro ele, em geral, perde o acesso à mesma, o qual passa a ser do receptor. A comunicação tem caráter assíncrono: mensagens partem de um processo origem e após um intervalo arbitrário de tempo chegam ao processo destino. Cada processo possui a capacidade de armazenar mensagens a ele destinadas, o que viabiliza o assincronismo e habilita o tratamento das mensagens em uma determinada ordem - que pode ser a de chegada. Os processos receptores devem explicitamente receber as mensagens. Não se requer que os processos estejam em um mesmo espaço de endereçamento, diferentemente da comunicação por memória comum. O único vínculo existente entre dois processos comunicantes é a mensagem que transita entre eles (Ruggiero et al., 1981). Isto tem efeitos positivos nas aplicações de sistemas distribuídos: como os processos são disjuntos quanto ao espaço de endereços, eles podem ser concebidos independentemente de equipamentos e repartidos posteriormente em várias máquinas de arquiteturas heterogêneas, sem alteração da comunicação. Esta característica dá ao projetista de programas paralelos flexibilidade e liberdade de abstração. Além do mais, habilita o desenvolvimento do programa num ambiente hospedeiro de forma a não necessitar de alterações substanciais, quando da distribuição dos componentes e implementação em sistemas distribuídos reais.

A troca de mensagens não reflete necessariamente uma hierarquia de controle entre os comunicantes, embora seja possível o relacionamento hierárquico. Para que se efetue a comunicação é necessário que haja identificação dos envolvidos. Além de dados, a mensagem deve conter endereços dos processos destinatário e remetente. É tarefa do subsistema de suporte de comunicação o encaminhamento e transferência de mensagens aos seus destinos.

Alguns sistemas operacionais são orientados, quanto à comunicação, para troca de mensagens, em contraposição aos orientados para procedimentos. Lauer & Needham (1978) identificam uma dualidade entre ambos os casos e concluem que o sistema operacional orientado para mensagens é mais indicado para aplicações de tempo real, por razões de desempenho e disponibilidade do sistema resultante. Enslow (1978) também concorda com o ponto de vista de que em sistemas distribuídos a comunicação se realize conforme o mecanismo de troca de mensagens, que ele chama de "protocolo tipo mensagem".

No modelo, a comunicação entre processos de módulos distintos dá-se sempre por troca explícita de mensagens. A referência a processos externos é sempre feita por nome. Os mapeamentos nome-endereço e endereço-nome são transparentes aos processos comunicantes e executados pelo subsistema suporte de comunicação.

Sobre a comunicação interna a um módulo vale a pena abrir um parênteses, embora não esteja inteiramente dentro do escopo deste trabalho. Os mecanismos usados para comunicação e sincronização entre processos de um módulo são os mesmos estudados anteriormente: via memória comum compartilhada ou por troca explícita de mensagens. Um fator condicionante na escolha de qual esquema a usar é a existência de sistemas operacionais e/ou linguagens de programação que forneçam um, outro ou mesmo ambos os esquemas. Como visto, os sistemas operacionais orientados a mensagens são mais indicados para programação de aplicações de tempo real. Todavia, é recomendável a troca de mensagens internamente por questões de compatibilidade e uniformidade.

3.3.4 Sincronização entre Processos

Na seção que caracteriza a comunicação em SDCD's foram levantados brevemente alguns pontos no que diz respeito ao sincronismo entre processos comunicantes. O caso mais frequente é o de comunicações assíncronas: um processo envia uma mensagem para outro e continua

o seu processamento normal; o outro processo em determinado momento verifica se há mensagem a ele destinada. Caso positivo, faz acesso `a mensagem e a consome; caso negativo, continua o processamento normal. Comunicações assincronas aplicam-se a todas as combinações de componentes comunicantes.

Comunicações sincronas são aquelas em que o remetente somente prossegue o processamento quando o destinatário recebe a mensagem enviada. O destinatário continua o processamento após receber a mensagem que estava esperando. Caso típico ocorre quando 2 processos devem executar alguma função ao mesmo tempo. Um diálogo é estabelecido entre os processos para que um se informe do estado do outro e vice-versa. No exemplo analisado de SDCD's, muitas das funções executadas são dirigidas por evento, o que pode levar a comunicações cujas mensagens cheguem aos destinatários em instantes de tempo aleatórios. Ademais, quando existe um subsistema de comunicação intermediando os processos comunicantes, os tempos transcorridos na transferência de mensagens são também aleatórios. Assim o caráter da comunicação em sistemas distribuidos torna-se eminentemente não deterministico. Hoare (1978) chama a atenção para este ponto: conhece-se bem o mecanismo de mensagens passantes deterministicas, havendo inclusive sistemas operacionais e linguagens que têm o conceito embutido. Este mecanismo facilita a implementação de comunicação síncrona, porém é viável nos sistemas centralizados porque existe uma única facilidade conveniente e não há problemas com a transmissão de informação.

Na comunicação com mensagens passantes não deterministicas existem dois problemas básicos. No primeiro, o processo que tem uma mensagem a ele destinada necessita ser informado da chegada da mesma, que pode estar sendo esperada ou não. O segundo problema ocorre quando existe mais de uma mensagem destinada a um processo. Há necessidade de se prover o armazenamento das mesmas até que o processo lhes dê tratamento. No modelo proposto considera-se que existam estruturas de filas gerenciadas pelo subsistema de suporte à comunicação para armazenamento das mensagens. A informação sobre chegada de mensagens será vista adiante.

À sincronização entre processos quando se usa mensagens passantes pode ser apresentada de forma sistemática, onde se define 4 modos básicos, segundo Manning et al. (1980). Os modos são:

1. envio bloqueante recepção bloqueante;
2. envio não bloqueante recepção não bloqueante;
3. envio não bloqueante recepção bloqueante;
4. envio bloqueante recepção não bloqueante.

Por bloqueante entende-se a operação que suspende a execução do processo corrente até a ocorrência de uma condição ou uma ordem de prosseguimento. O bloqueio, no envio, perdura até que o remetente receba do destinatário um aviso informando da recepção da mensagem. Na recepção, o bloqueio perdura até chegar a mensagem solicitada.

O primeiro modo leva a comunicações totalmente sincronas, útil portanto na transferência de mensagens deterministicas e com atrasos de transmissão também deterministicos. Este modo garante a indivisibilidade da transmissão de mensagens. Uma só transferência tomará lugar de cada vez, o que não requer na implementação o enfileiramento de mensagens. Por outro lado, acarreta maior complexidade na implementação do subsistema de comunicação, que deve prover mecanismos para deteção de perdas irrecuperáveis de mensagens e conseqüente informação do fato aos processos comunicantes. Também requer um fluxo adicional de mensagens de controle da comunicação. Finalmente, este modo totalmente sincrono não permite o aproveitamento do potencial de paralelismo inerente aos sistemas distribuidos.

O segundo modo é totalmente assincrono e permite o tratamento de mensagens não deterministicas. Isto porque, se um processo não for bloqueado na recepção, ele é potencialmente capaz de, a qualquer instante, verificar se existe alguma mensagem a ele destinada e então recebê-la. Os processos envolvidos podem ser executados com o máximo de paralelismo possivel, caracteristica desejável nas aplicações de controle distribuido em tempo real. Todavia requer o gerenciamento de filas de mensagens em trânsito.

Neste caso convém ressaltar que as filas podem assumir, mesmo que temporariamente, tamanhos arbitrariamente grandes, o que não condiz com uma implementação real onde existe um limite superior na capacidade de memória do sistema. Assim, mecanismos devem ser previstos para tratar dessa situação, dos quais se pode sugerir o descarte de mensagens e o estabelecimento de limites no número de mensagens que um processo pode estar enviando num momento ("janela").

O terceiro modo é semi-síncrono, requer gerenciamento de filas mas permite parcialmente execuções paralelas. Existem situações onde é desejável, como no caso de um processo estar esperando uma ordem para prosseguimento. O quarto modo apresenta as desvantagens de implementação do primeiro, embora não tenha muitas aplicações.

A sugestão para incorporação desses modos de sincronização no modelo de programação paralela é que ele deve contar com o envio não bloqueante/recepção bloqueante e envio não bloqueante/ recepção não bloqueante. Estas discussões consideram a sincronização implícita através de um par envia/recebe, que é resolvida pelo suporte à comunicação. Porém dois processos podem sincronizar-se de maneira explícita, através da composição dos dois modos sugeridos.

Existe uma outra alternativa para o bloqueio na recepção. A idéia é que ele seja feito por um intervalo de tempo arbitrário, a critério do programador da aplicação. Findo o intervalo especificado, se a mensagem requerida não foi recebida, o processo continua o processamento normal (mecanismo de "time-out"). Requer entretanto uma complexidade a mais na implementação. A vantagem desse modo é que evita a ocorrência do problema do bloqueio por tempo indeterminado (ou perpétuo) e habilita a liberação de recursos computacionais para outros requisitantes, no caso de sistemas com multiprogramação e/ou multitarefagem.

3.3.5 Esquemas de Endereçamento

Na comunicação com mensagens passantes os processo que se comunicam devem explicitar os destinatários e remetentes de mensagens. Este aspecto é chamado de endereçamento ou nomeação. Foram vistas várias combinações de componentes que se comunicam em um SDCD. Por exemplo, um processo de um módulo que implementa o SMO envia mensagens a um ou mais processos de módulos que implementam ECL's. Uma ECL pode se comunicar com o SMO e com outras ECL's. Existem 4 esquemas básicos de endereçamento de processos, sendo 2 para processos transmissores e 2 para processos receptores de mensagens:

1. envio único - um transmissor para um receptor;
2. envio múltiplo - um transmissor para vários receptores;
3. recepção única - um receptor recebe de um transmissor;
4. recepção múltipla - um receptor recebe de vários transmissores

O envio múltiplo permite que uma mensagem seja enviada a vários destinatários em uma só operação. Tem-se o caso de envio para todos processos do programa ("broadcasting") ou o caso do envio seletivo, onde é especificada uma lista de processos destinatários. Estas duas opções são confortáveis do ponto de vista do programador mas trazem problemas para a implementação: o controle de uma transmissão confiável fica complexo pelo fato de serem vários os processos envolvidos e o campo de endereçamento na mensagem passar a ser de tamanho variável. O envio tipo um-para-todos ocorre eventualmente em sistemas distribuidos e, segundo Gentleman (1981) este modo não encontra uso que justifique sua implementação. Entretanto, no caso de aplicações de controle de processos, este esquema se apresenta muito interessante nas inicializações, reconfigurações e recuperações de estado de falha. O envio múltiplo pode ser obtido indiretamente com a emissão de uma série de envios únicos.

A recepção múltipla, com origem não especificada, é fundamental para se implementar não determinismos com relação à origem das mensagens. Seja um caso em que uma ECL A mantém um diálogo com

uma ECL B e apenas envia mensagens ao SMO, num esquema ciclico. A ECL A emitiria envios especificos ao SMO e recepções especificas da ECL B. Se, por uma razão qualquer, o SMO necessitasse enviar uma mensagem à ECL A, esta não teria como recebê-la. Adotando a recepção múltipla este problema fica solucionado: após receber uma mensagem e identificar sua origem, a ECL A tomaria decisões convenientes.

No modelo de programas paralelos o projetista deve contar com os esquemas de envio especifico e recepção especifica e não especifica. A opção de se eliminar o envio múltiplo visa a simplicidade de implementação e eficiência do sistema de suporte à comunicação. Mesmo assim vale a pena discutir alguns aspectos. Quando a sub-rede apresenta facilidades de difusão múltipla, o envio múltiplo fica simples: uma mesma mensagem gerada em um nó é entregue a cada um dos nós restantes. Porém, na entrega de mensagens aos processos destinatários, é preciso ter mecanismos de replicação da mensagem nas dependências do hospedeiro. Quando a sub-rede não apresenta facilidades de difusão múltipla, existem duas opções. Na primeira uma mensagem é replicada e enviada para cada nó. A entrega aos processos destinatários é feita da mesma maneira anterior. Na segunda opção uma mensagem é replicada e enviada especificamente a cada processo destinatário. Neste caso, a entrega aos processos nas dependências do hospedeiro é simples.

Um ponto ainda a complementar o modelo diz respeito ao tratamento prioritário de mensagens. Pode ser interessante para a aplicação que o suporte à comunicação trate mensagens com prioridades diferenciadas. No modelo considera-se a existência de classes de mensagens para efeito de atendimento. Como sugestão define-se 3 classes:

1. mensagens de controle;
2. mensagens de dados urgentes;
3. mensagens de dados normais

3.4 SUBSISTEMA PARA COMUNICAÇÃO DA APLICAÇÃO

A presente seção trata da especificação de um subsistema para suporte à comunicação entre processos computacionais. Em particular entre processos que implementam as aplicações de controle em um SDCD, simplesmente chamados de processos da aplicação. Este subsistema de suporte é visto como um nível de serviço da arquitetura da rede e deve obedecer a um protocolo de comunicação. Os problemas existentes na concepção de software de comunicação são em grande parte semelhantes ao da concepção de outros tipos de software. Porém há que se considerar alguns pontos especiais, como a necessidade de compatibilização entre diferentes componentes, às vezes heterogêneos; a possibilidade de implementação por grupos distintos e em locais também distintos; a dificuldade de entendimento do comportamento geral do sistema, devido à existência de operação paralela dos vários componentes (Bochmann, 1983). Assim torna-se relevante a especificação precisa e a verificação do comportamento dos diversos componentes do sistema.

A especifição de um determinado nivel (n) é dividida em duas partes: a primeira é a <u>especifição de serviço</u>, que define o comportamento geral do subsistema nivel (n) - um subsistema nivel (n) se compõe do nível (n) e dos niveis inferiores. A segunda parte é a <u>especificação de protocolo</u>, que define o comportamento dos componentes de um mesmo nivel. Estes componentes são conhecidos como <u>entidades</u> (Ver a figura 3.1). Cada entidade de um mesmo nivel está usualmente localizada em um elemento da rede. Através de cooperação entre si, elas provêm o serviço especificado. Para tal usam os serviços do nivel imediatamente inferior.

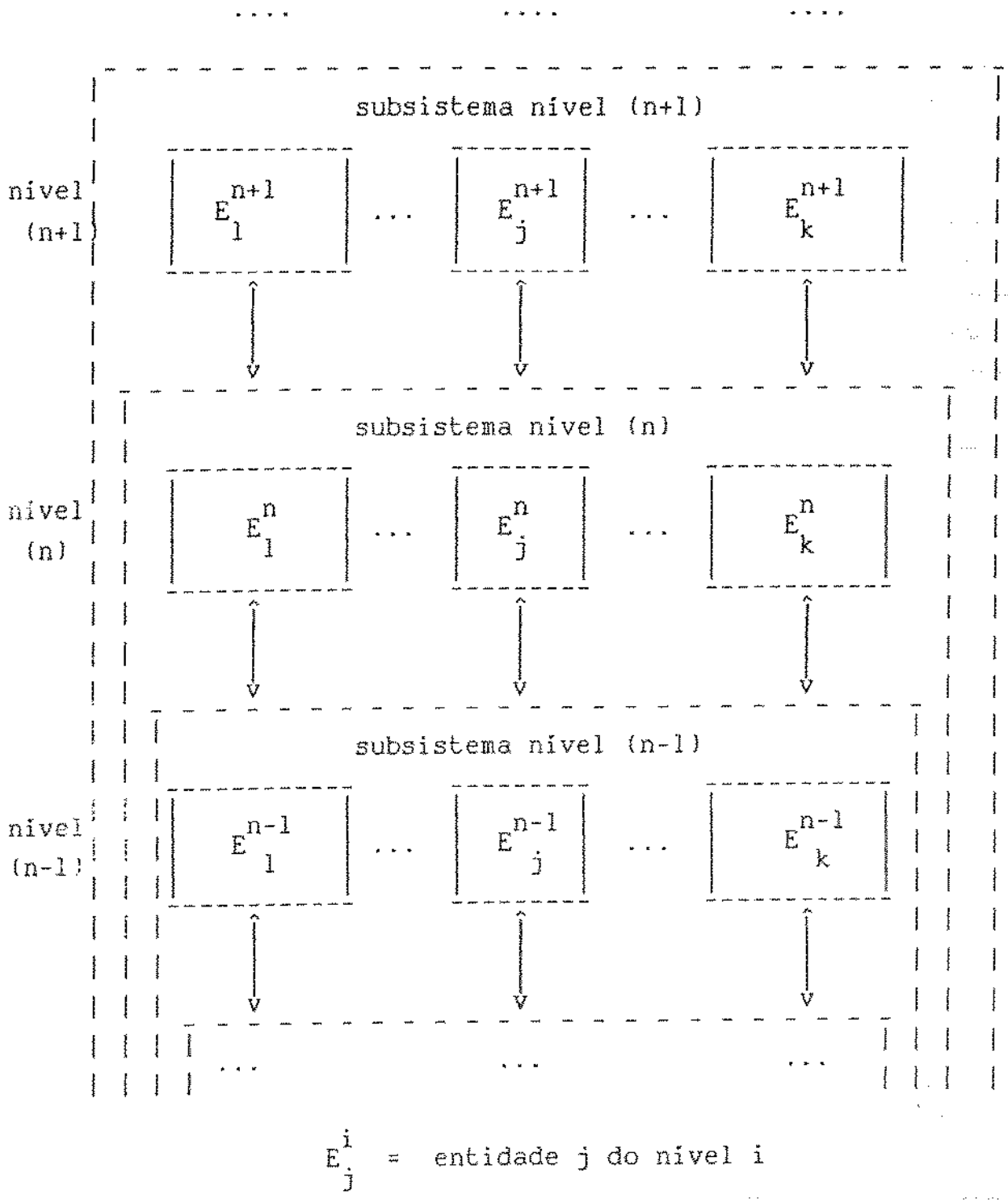


Figura 3.1 Estrutura de um subsistema de comunicação

Antes porém de partir para a especificação, desenvolve-se, a título de exemplo, um modelo de referência que mostre, no escopo de SDCD, os níveis envolvidos. Um sistema de controle SDCD como um todo, no sentido da comunicação, tem sua arquitetura apresentada na figura 3.2. Esta se compõe de uma série de subsistemas, estruturados em forma de camadas envolventes. Cada subsistema se apóia no seu vizinho interior (subsistema envolvido) para efetuar a comunicação entre suas entidades. O sistema como um todo, ou o subsistema mais externo, é o próprio SDCD. Suas entidades têm como função executar o controle do processo físico e são conhecidas coletivamente como APLICAÇÃO, abreviadamente, APLIC. Para tal devem se comunicar entre si usando os serviços do próximo subsistema envolvido. As entidades de APLIC são aqueles componentes de um SDCD vistos no capítulo anterior:

.módulos do subsistema de monitoração e operação (SMO);
.módulos do subsistema de supervisão e otimização (SSO);
.módulos do subsistema de controle local (SCL), constituído pelas estações de controle local (ECL).

O subsistema seguinte oferece os serviços a serem definidos, denominado subsistema de suporte à comunicação da aplicação (SCOM). O nível correspondente é designado de SUCO, constituído das entidades ESUCO, objetivo desta especificação.

O subsistema inferior SINTX, de Subsistema de Interconexão, provê serviços de interconexão das entidades ESUCO. Neste trabalho ele é visto como um subsistema terminal que provê serviços de transporte de informação entre os vários sítios que constituem a rede, uma vez que não é de interesse nem necessário para os propósitos deste trabalho, a decomposição do mesmo em outros subsistemas sucessivos. Usualmente, em sistemas reais, o SINTX é implementado como mais de um nível de serviço. Todavia a identificação do perfil de serviços do subsistema é necessária, como será feito adiante.

A título de se situar perante as padronizações da ISO e do Comitê 802 do IEEE, o SINTX implementa os níveis físico, de enlace e de

rede, este último apenas para o padrão da ISO.

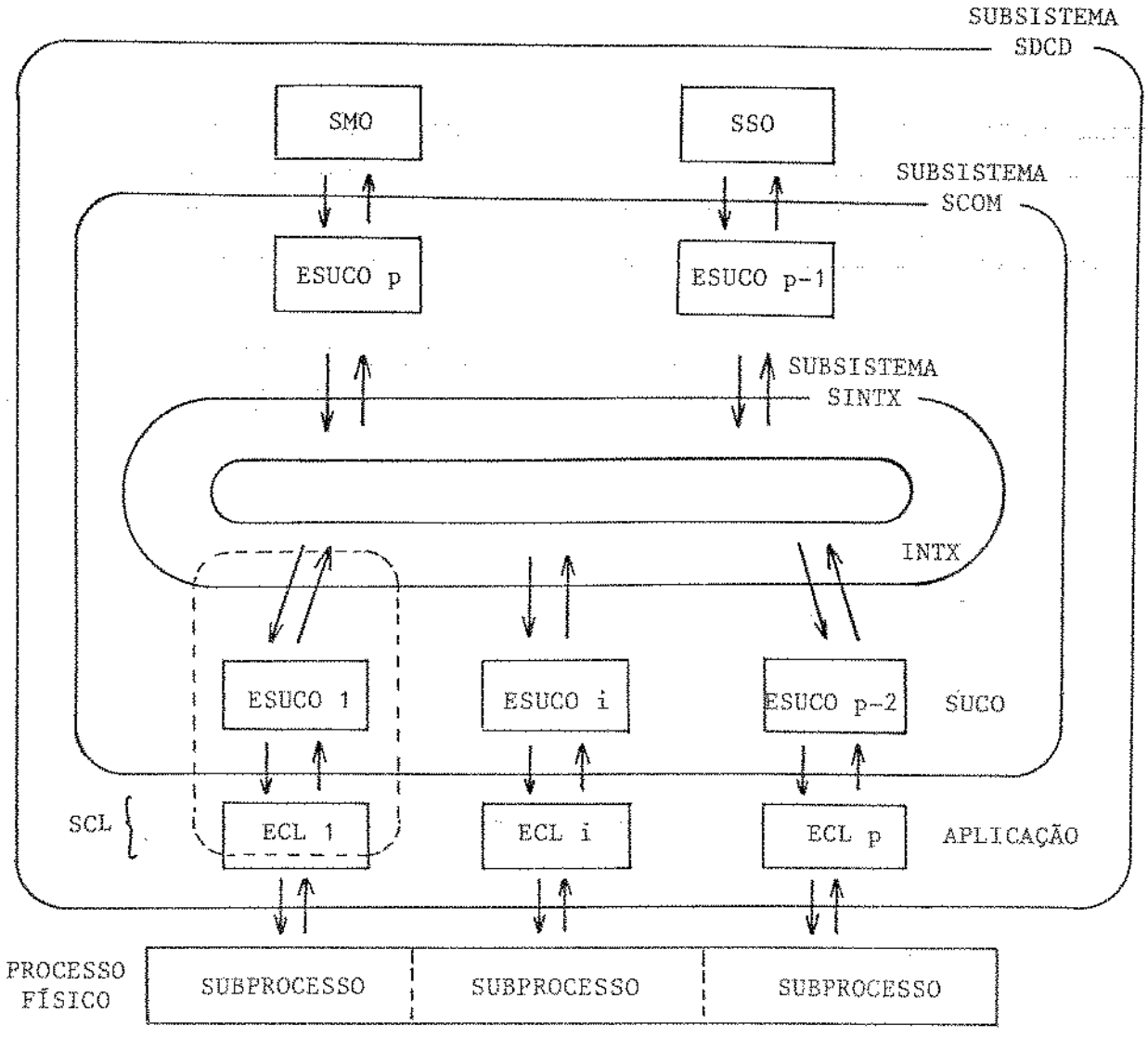


Figura 3.2 Modelo de Arquitetura de Interconexão de um SDCD

3.4.1 Especificações de Serviço

3.4.1.1 Serviços do Subsistema de Interconexão

O nível superior INTX do subsistema de interconexão, doravante abreviado para INTX, implementa os níveis inferiores da arquitetura de interconexão - nível (n-1) da figura 3.1. Ele provê um serviço de comunicação para o nível SUCO ou nível (n), que consiste no transporte de mensagens entre os nós de rede remotamente situados, na modalidade "datagrama", conforme definido na seção 2.2.6. A seguir descrevem-se as características e funções requeridas e/ou esperadas para o nível INTX.

Características gerais

Mensagens são entregues para o nível superior livres de erro;

Não há sequenciamento de mensagens, ou seja, a ordem de entrega não é necessariamente a de recepção;

INTX gera mensagens e as entrega ao SUCO - nível (n); INTX consome mensagens oriundas do SUCO;

Existe controle de fluxo de mensagens para evitar sobrecarga do subsistema de interconexão

A entrega de mensagens é provida pela primitiva recebe, emitida pelo nível SUCO;

A recepção de mensagens é provida pela primitiva envia, emitida pelo nível SUCO;

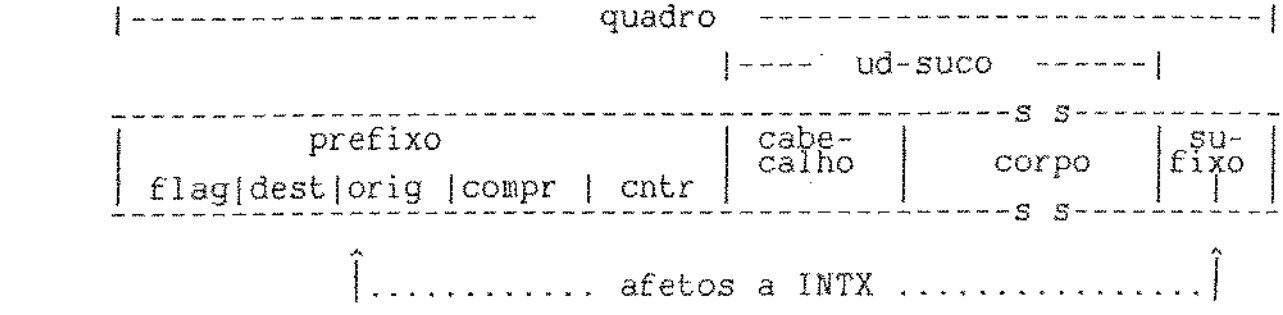
Os parâmetros das primitivas configuram uma mensagem, que é de tamanho variável. Esta compõe-se de um cabeçalho e um corpo de dados, como esquematizado abaixo;

O cabeçalho e o corpo de dados não têm significado especial para INTX e são considerados conjuntamente como uma sequência de bits a serem transmitidos.

A unidade de informação que o INTX se encarrega de transportar é chamada de quadro. Este compõe-se de um prefixo, da mensagem e de um sufixo;

O prefixo contém endereços dos nós de rede destino e origem da mensagem, o comprimento da mensagem, um campo de controle e um campo de "flag" para indicação de início de quadro;

O sufixo contém informação para controle de erro de transmissão (CRC, paridade, etc) e um "flag" para indicação de final de quadro;



Primitivas de comunicação

Por primitivas de comunicação de um determinado nível se entende como as operações básicas executadas pelas entidades desse nível, para prover os serviços necessários ao nível superior. Ademais, as primitivas definem a interface entre esses dois níveis adjacentes. A interface entre INTX e o nível superior é implementada pelas primitivas envia e recebe, no caso emitidas pelo nível superior, da seguinte maneira:

envia

---> nó destino
---> comprimento
---> mensagem

recebe

<--- nó origem
<--- comprimento
<--- mensagem

onde o sentido da seta mostra o sentido de transferência do parâmetro, do ponto de vista do chamador da primitiva. Uma seta da esquerda para a direita indica que o chamador fornece o valor do parâmetro. Uma seta no sentido contrário indica que o chamador recebe o valor do parâmetro ao fim da operação.

Os parâmetros nó origem e nó destino assumem valores correspondentes aos endereços dos processos origem e destino, em termos dos elementos de computação, conectados à rede de interconexão (nós de rede).

O parâmetro comprimento deve ter um valor mínimo correspondente ao comprimento do cabeçalho, para casos em que o corpo seja nulo. O valor máximo depende de implementação.

O parâmetro mensagem é uma referência à cadeia de bits formada pelo cabeçalho mais o corpo. Assume valores usuais de endereçamento, dependentes de implementação.

Sincronização e sequenciamento

A ordem de execução de uma mesma primitiva é sequencial, ou seja, uma nova chamada da primitiva, por uma entidade do nível superior, somente pode ser feita após o término da chamada anterior. A política de entrega de mensagens é exclusiva de INTX.

As primitivas envia e recebe podem ser chamadas pelo nivel superior, independentemente uma da outra.

Propriedades quantitativas

Uma caracteristica importante dos serviços prestados por uma camada são as propriedades quantitativas de vazão suportada e de atraso imposto às mensagens transportadas. Como o objetivo desta especificação do nivel (n-1) é dar subsidios para a especificação do nivel (n), os valores são importantes para se saber a capacidade oferecida. Além disso, estes valores são interessantes para se proceder a análises de desempenho, por simulação ou por ferramentas matemáticas. Este tema será tratado num capitulo posterior deste trabalho. Para o nivel INTX seguem-se as propriedades.

Vazão: INTX é capaz de produzir/consumir, segundo Jensen (1978), entre 100 a 1000 msg/s com tamanhos médios de mensagem entre 10 e 100 palavras de 8 bits. Existe uma relação inversa entre a taxa de produção/consumo e o comprimento da mensagem. Um sistema típico em anel, apresentado por Schoeffler (1984), transportando mensagens de comprimento médio igual a 90 bytes, pode processar 700 mensagens por segundo. Do ponto de vista de uma entidade, esta capacidade será reduzida, pois há que se considerar a existência de outras entidades como fornecedoras de mensagens. Considerando, por exemplo, uma rede com 10 entidades fornecendo mensagens à mesma taxa, esta deveria ser, em média, de 70 msg/s para que o subsistema de comunicação, no total, processasse 700 msg/s.

Atraso: para uma taxa de produção/consumo de 500 msg/s e tamanho médio de 500 bits/msg, em modelos típicos da literatura (Lages, 1981; Liu, 1978), o atraso médio imposto pelo subsistema de interconexão é da ordem de 2 a 5 ms. A rede local de Cambridge, conforme Lunn & Bennet (1981),

apresenta atrasos da ordem de 6 a 9 ms para taxas de 1000 msg/s e 320 bits/msg. Segundo Steusloff (1984) os atrasos são da ordem de 1 a 10 ms. Estes valores referem-se apenas ao transporte físico das mensagens. Quando se considera os protocolos de comunicação, em geral implementados por software, o atraso deve se elevar: uma transmissão fim-a-fim confiável impõe atrasos devido ao surgimento de mensagens de reconhecimento e eventuais retransmissões. Ainda segundo Schoeffler, para o mesmo exemplo do anel citado acima, o atraso total fica da ordem de 10 a 15 ms, para sistemas considerados grandes e altamente carregados.

3.4.1.2 Serviços do Subsistema de Comunicação - SCOM (nível SUCO)

O nível de suporte à comunicação da aplicação (SUCO), corresponde ao nível (n) da arquitetura de interconexão da figura 3.1. Ele provê serviços de comunicação para o nível (n+1) - APLIC, e usa os serviços providos pelo nível (n-1) - INTX, definidos na seção anterior. Os requisitos a serem atendidos para o nível de aplicação foram informalmente descritos na seção precedente, de número 3.3, que trata de programação paralela. Nesta seção objetiva-se especificar os serviços providos pelo SUCO.

Características gerais

A unidade de informação tratada neste nível é denominada unidade de dados do SUCO ou ud-suco;

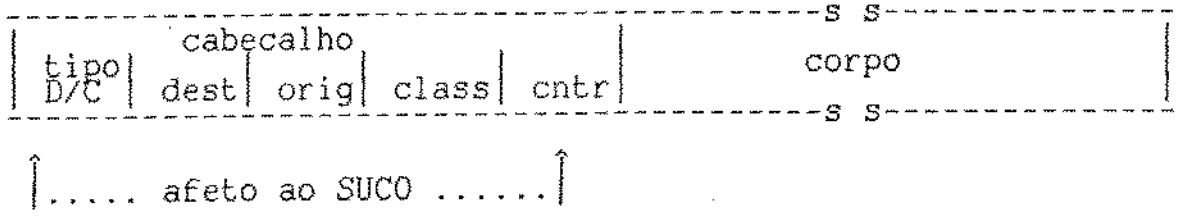
O SUCO recebe mensagens oriundas de INTX e entrega mensagens para INTX; O SUCO recebe mensagens oriundas de APLIC e entrega mensagens para APLIC;

As mensagens entregues a APLIC podem ser originárias de INTX ou do próprio SUCO; As mensagens entregues a INTX podem ser originárias de APLIC ou do próprio SUCO;

As mensagens recebidas pelo SUCO podem ser passadas adiante

ou consumidas pelo próprio SUCO;

As mensagens processadas pelo SUCO são de tamanho variável e compostas de um cabeçalho e um corpo de dados. O cabeçalho é de tamanho fixo e contém o tipo de mensagem (dados ou controle), os nomes ou números dos processos origem e destino, a classe da mensagem para efeito de atendimento e um campo de função de controle. O corpo é de tamanho variável, com existência opcional. O esquema de um ud-suco segue abaixo;



A interface entre SUCO e INTX foi definida na seção anterior.

Primitivas de comunicação

O SUCO provê para a aplicação o seguinte conjunto de operações primitivas, que são chamadas do nível APLIC:

envia-mensagem
envia-resposta
recebe-mensagem
recebe-resposta
procura-mensagem;

Os parâmetros de cada primitiva são apresentados na tabela a seguir, onde se indica também o sentido de transmissão dos mesmos, seguindo as mesmas convenções do item anterior. Em seguida vem uma explanação sobre os possíveis valores que podem assumir.

nome da primitiva/parâmetros	observações
envia-mensagem e envia-resposta ---> processo destino ---> processo origem ---> comprimento ---> classe ---> mensagem <--- sucesso	destino especifico não bloqueante
recebe-mensagem <--- processo origem ---> processo destino <--- comprimento <--- classe ---> tempo de espera <--- mensagem <--- sucesso	origem não especifica bloqueante
recebe-resposta ---> processo origem ---> processo destino <--- comprimento <--- classe <--- mensagem ---> tempo de espera <--- sucesso	origem especifica bloqueante
procura-mensagem <--- processo origem ---> processo destino <--- comprimento <--- classe <--- mensagem <--- sucesso	origem não especifica não bloqueante

Os parâmetros processo origem e processo destino assumem valores dentro do conjunto de nomes de processos endereçáveis. O parâmetro comprimento é um inteiro correspondente ao número de bytes ou octetos da mensagem, que vai de zero a um máximo dependente de implementação. O parâmetro classe é um inteiro que assume valores dentro do número de classes de prioridade a que as mensagens podem pertencer. O parâmetro mensagem é uma referência ou apontador para um "buffer" que contém os dados que constituem a mensagem. O parâmetro tempo de espera especifica, em unidades de tempo, o maior intervalo que o processo emitente da primitiva pode ficar bloqueado à espera da mensagem pedida. O parâmetro sucesso indica se a operação foi bem sucedida, ou seja, se a mensagem pedida está sendo entregue ou não.

Regras de sincronização e sequenciamento

A ordem de execução das primitivas de envio emitidas por um processo da aplicação é sequencial, ou seja, uma nova chamada só pode ser feita após o término da chamada anterior. Neste sentido, que diz respeito à sincronização entre as entidades do SUCO e os processos de aplicação, o envio é uma operação bloqueante, que só termina quando o SUCO se apodera da mensagem;

A ordem de execução das primitivas de recepção por um processo da aplicação é sequencial. As operações são bloqueantes por tempo determinado, ou seja, se houver mensagem o SUCO procede a entrega; caso contrário bloqueia o processo até que a mensagem pedida chegue ou o tempo de bloqueio se esgote. A entidade ESUCO que executa a primitiva porém não fica bloqueada durante esse tempo;

A ordem de execução da primitiva procura-mensagem é sequencial e não bloqueante. Se houver mensagem ela é entregue; caso contrário o processo de aplicação é notificado e liberado;

Para um mesmo processo de aplicação, a execução de quaisquer das primitivas é sempre sequencial. A primitiva recebe-resposta de um processo especificado deve ser executada após

uma envia-mensagem para o mesmo processo. A primitiva envia-resposta para um determinado processo deve ser executada após uma recebe-mensagem ou procura-mensagem referente ao mesmo processo.

Regras de comportamento fim-a-fim

Para cada envia-mensagem executada, deve ocorrer no processo destinatário a execução bem sucedida de uma recebe-mensagem ou procura-mensagem, na qual a mensagem recebida seja aquela enviada pelo remetente;

Para cada envia-resposta executada, deve ocorrer no processo destinatário a execução bem sucedida de uma recebe-resposta ou uma procura-mensagem, na qual a mensagem recebida seja aquela enviada pelo remetente.

Propriedades quantitativas requeridas

O subsistema de suporte impõe um atraso não nulo às mensagens que por ele transitam. Para determinar o atraso máximo permitido, há que se considerar como fator limitante o atraso permitido pela aplicação e como fator restritivo o atraso intrinseco do subsistema de interconexão. Denota-se As o atraso médio imposto pelo SUCO, Ai o atraso médio imposto pela interconexão e Aa o atraso médio permitido pela aplicação. Numa transmissão em que uma mensagem vai de uma entidade da aplicação à outra remotamente situada, essa deve ser processada pelo SUCO, pela interconexão e novamente pelo SUCO. Assim,

$$Aa >= Ai + 2\,As$$

ou

$$As =< (Aa - Ai) / 2$$

Valores de vazão e atraso serão considerados num capitulo posterior, quando de estudos de desempenho. Deve-se ressaltar, entretanto, que o processamento de mensagens a nivel de SUCO é feito a altas taxas e com pequenos atrasos, se comparados com as grandezas envolvidas na aplicação e no

subsistema de interconexão. Isto porque este processamento é feito no âmbito de processadores e memórias locais.

3.4.2 Especificação do Protocolo

O protocolo de comunicação entre as entidades do nível SUCO é bastante simples, baseado nas idéias de comunicação através de "datagramas".

Esta especificação descreve a operação das entidades do SUCO em resposta a comandos da aplicação, do subsistema de interconexão e de outras entidades.

As entidades do SUCO recebem comandos da aplicação através das primitivas anteriormente citadas (envia, recebe e procura). Cada entidade mantém um conjunto de estruturas de filas para acomodar as mensagens em trânsito no correspondente nível (n). Também mantém uma tabela que contém os nomes de todos processos endereçáveis e o correspondente nodo de rede em que cada um se encontra.

Ao executar uma das primitivas envia-mensagem ou envia-resposta, uma entidade do SUCO toma as seguintes atitudes:

verifica o destino da mensagem;

se o destino for remoto, processa a mensagem e a encaminha para o exterior, ou seja, para o nível (n-1);

se o destino for local, encaminha a mensagem para o processo destinatário, nível (n+1);

se o destino for a própria entidade, encaminha a mensagem para ser consumida localmente;

se não for possível identificar o destino, significa que o destinatário não existe. Neste caso notifica o remetente.

Ao executar a primitiva recebe-mensagem, uma entidade do SUCO toma as seguintes atitudes:

inicia contagem do tempo máximo de espera especificado;

verifica se há uma mensagem destinada ao processo solicitante;

se houver, procede à entrega;

se não houver, continua a contagem de tempo até que uma mensagem chegue ou o tempo se esgote. No primeiro caso, procede à entrega. No segundo caso, notifica o solicitante;

a politica de entrega prioriza as mensagens segundo suas classes. Dentro de uma classe a entrega segue a politica FCFS (de "First Come First Served").

Ao executar a primitiva recebe-resposta, uma entidade do SUCO toma as seguintes atitudes:

inicia contagem de tempo máximo de espera especificado;

verifica se há alguma mensagem destinada ao solicitante, a qual tenha o remetente coincidindo com o especificado;

se houver, procede à entrega;

se não houver, continua contagem de tempo até que uma mensagem chegue ou o tempo se esgote. No primeiro caso, procede à entrega. No segundo, notifica o solicitante.

Ao executar a primitiva procura-mensagem, uma entidade do SUCO toma as seguintes atitudes:

verifica se há alguma mensagem destinada ao solicitante;

caso positivo, procede à entrega. Caso negativo, notifica o solicitante;

a politica de entrega prioriza as mensagens segundo suas classes. Dentro de uma mesma classe a politica é FCFS.

Para envio de uma mensagem ao exterior através de INTX, o SUCO comanda a execução da primitiva envia do nível INTX. A politica de entrega prioriza as mensagens segundo suas classes. Dentro de uma classe a politica é FCFS.

Para recepção de mensagens oriundas do exterior através de INTX, havendo espaço em suas estruturas internas de armazenamento, o SUCO comanda a execução da primitiva recebe do INTX. Ao receber uma mensagem do exterior, o SUCO toma as seguintes atitudes:

verifica o destino da mensagem;

se o destino for um processo de aplicação, encaminha a mensagem ao mesmo. Se este estiver esperando pela mensagem recém chegada, procede à entrega. Caso contrário, armazena a mensagem;

se o destino for a própria entidade, encaminha a mensagem para ser consumida localmente pelo SUCO;

se o destino não for identificado, significa que o processo destinatário não pertence ao nodo corrente. Neste caso a mensagem é encaminhada para ser consumida localmente, o que posteriormente gerará uma nova mensagem, notificando o fato, para ser enviada ao remetente.

As mensagens recebidas pela entidade para serem consumidas localmente trazem como informação principal o estado da rede e dos processos de aplicação. Nesses termos, mensagens recebidas de outras entidades implicam na alteração do conteúdo da tabela de nomes e endereços de processos.

Uma entidade do SUCO deve executar as diversas operações de forma concorrente, de modo a não ficar bloqueada enquanto as operações não se completam. Por exemplo, pode estar esperando uma mensagem de INTX ao mesmo tempo que está entregando uma mensagem a um processo de aplicação.

3.5 ESPECIFICAÇÃO DE IMPLEMENTAÇÃO

Esta seção visa a descrição mais detalhada do subsistema anteriormente especificado e a adição de um formalismo na especificação, para facilitar tanto implementações reais como validações das entidades. Os detalhes adicionais incluem:

os aspectos de comportamento de uma entidade não tratados anteriormente;

os aspectos da estrutura interna de uma entidade;

detalhes sobre as interfaces;

algoritmos.

A especificação de implementação dar-se-á primeiramente considerando uma entidade do SUCO e seu ambiente. Este é composto, de um lado, do nível provedor de serviço (INTX) e, de outro lado, do nível usuário de serviço (APLIC). Na figura 3.2, seção anterior, os elementos envolvidos estão mostrados dentro de uma linha pontilhada.

Para complementar as informações da seção anterior de especificação, apresenta-se na figura 3.3 um esquema simplificado mostrando o fluxo de mensagens fluindo para/de uma entidade do SUCO e às principais estruturas de dados (filas).

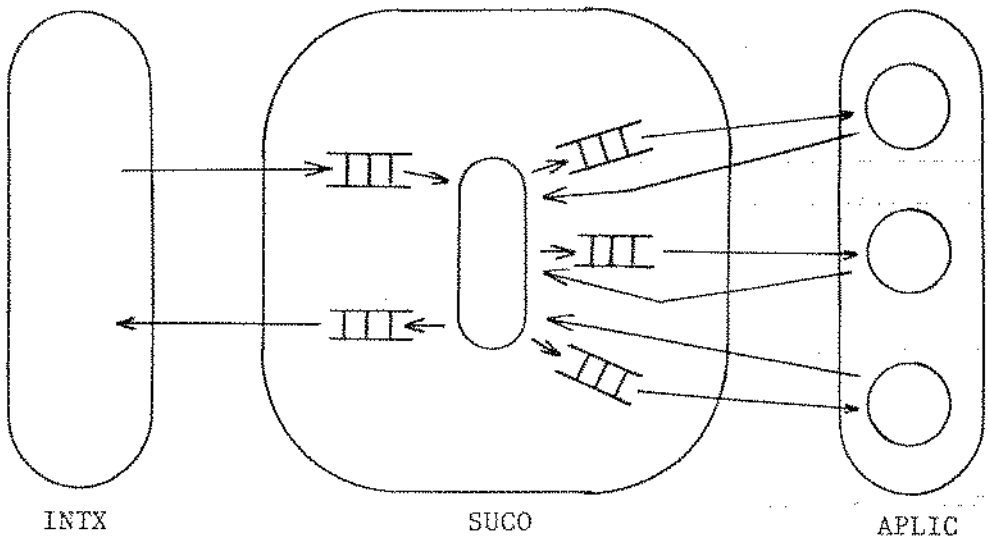


Figura 3.3 Fluxo de mensagens e filas em uma entidade do SUCO.

3.5.1 A Metodologia de Descrição

A descrição das entidades e protocolos do SUCO, com mais detalhes e com formalismo, será feita usando a metodologia embutida no NUCA - Núcleo de Concepção Arquitetônica, um sistema de PAC de Sistemas Digitais desenvolvido no Departamento de Ciência da Computação - UFMG (Paula Filho & Campos, 1981; Arabe & Campos, 1981). Este sistema de PAC abriga ferramentas de modelagem e simulação, mas o que interessa em particular é apenas a modelagem tanto da estrutura como do comportamento de um programa em concepção. A metodologia vem sendo usada com sucesso em vários projetos do Departamento.

A concepção, segundo a metodologia, consiste na modularização de um sistema digital em passos descendentes de refinamento. Em cada passo identifica-se um conjunto de módulos interligados, obtendo-se assim um modelo estrutural, e associa-se a cada módulo um modelo de seu comportamento. Cada novo passo consiste em refinar um (ou mais) módulo(s), da mesma forma identificando novos conjuntos de módulos nele contidos e associando-lhes modelos comportamentais. O processo de concepção termina quando

todos os módulos estiverem descritos em formas implementáveis. Obviamente o grau de detalhamento requerido dependerá dos recursos disponíveis à implementação: nível da linguagem adotada e/ou grau de integração dos componentes eletrônicos.

O primeiro passo a ser dado consiste na apresentação do universo do projeto, composto do subsistema a ser projetado e do ambiente conhecido.

Os elementos para descrição estrutural são módulos, interconexões e soquetes. Os módulos representam subsistemas capazes de desempenhar processos e/ou armazenar dados, e se comunicam com seu ambiente por meio de interfaces bem definidas. Os soquetes modelam as interfaces de interligação dos módulos, a que são associados, com o seu ambiente. As interconexões indicam a interligação de dois ou mais módulos, modelando os caminhos de transferência de dados e/ou controle (Ver a figura 3.4). A posição do nome do soquete, se situado internamente ou externamente ao módulo, informa se o elemento associado ao soquete é definido dentro ou fora do módulo, respectivamente.

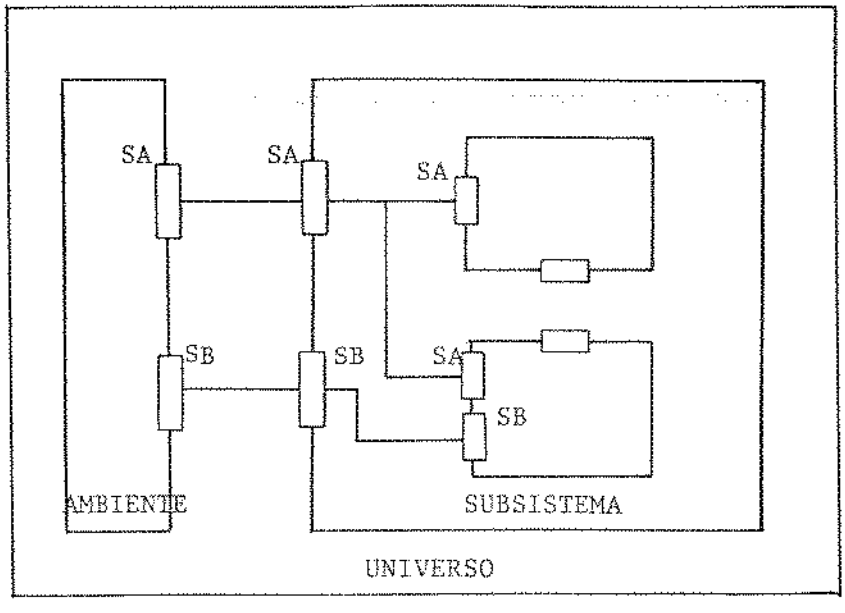


Figura 3.4 Ilustração de modelagem estrutural.

A descrição do comportamento inerente a cada módulo dá-se por um

grafo de controle, cujos elementos são nós, arcos de controle e marcadores (Ver a figura 3.5). Os nós, representados por círculos, indicam a existência de algum processamento, e são associados a trechos sequenciais de algoritmos que modelam o comportamento do módulo. Os arcos de controle, representados por arcos orientados, indicam o fluxo de controle entre nós e definem as relações de precedência temporal. Os marcadores, representados por círculos cheios, são indicadores de linhas ativas de controle. Cada nó tem em sua entrada um conjunto de arcos de controle relacionados por uma expressão lógica de entrada, que especifica as condições para que o nó seja ativado. Na saída de cada nó há uma expressão lógica de saída dos arcos que aí se originam, onde é especificado quais arcos ficarão ativos (receberão marcadores) ao final do processamento do nó.

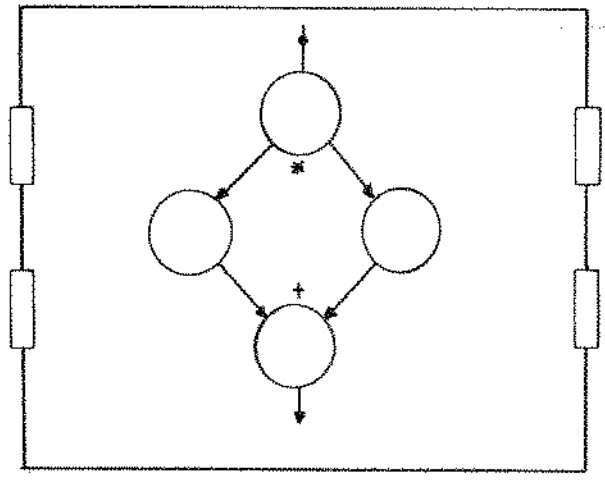


Figura 3.5 Modelagem de comportamento

A descrição do comportamento conta ainda com uma declaração das variáveis locais ao módulo e, portanto, acessíveis internamente pelos algoritmos dos nós. Além disso, são declarados os soquetes e especificados seus atributos, desse modo definindo a interface com o meio externo.

Algumas convenções foram adicionadas para permitir o modelamento de procedimentos reentrantes e tipos abstratos de dados, ampliando assim a aplicabilidade da ferramenta (Nogueira, 1984).

Procedimentos são conhecidos externamente por soquetes definidos como "ponto de entrada". Estes soquetes modelam a passagem de controle, quando da chamada e retorno, e descrevem os parâmetros. Um procedimento, ao ser ativado (chamado), tem disponiveis no soquete os parâmetros que foram definidos como sendo de entrada. Quando um procedimento retorna, o chamador tem disponiveis no soquete os parâmetros da saida. A chamada de procedimentos é indicada de duas maneiras: através de indicação explicita no grafo de controle e através de referência ao nome do procedimento na descrição algoritmica dos nós, como mostrado na figura 3.6.

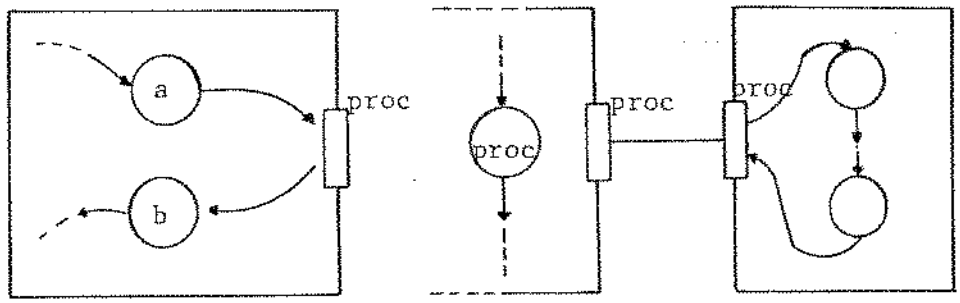


Figura 3.6 Chamada de procedimentos no NUCA

O modelo permite a descrição de procedimentos reentrantes, e esconde do projetista os mecanismos de controle de chamada, retorno, passagem de parâmetros e criação de instâncias de código. Na modelagem que se segue, sempre que necessário serão feitos comentários explicativos acerca da metodologia. Deve-se chamar a atenção que a metologia é uma boa ferramenta de documentação de projeto.

3.5.2 Modelo Estrutural do SUCO e seu AMBIENTE

Na figura 3.7 tem-se a estrutura do sistema em descrição, mostrada em seu nivel mais alto de abstração. Antes convém esclarecer que o termo módulo faz parte da metodologia, e não tem necessariamente relação com os módulos do modelo de programas

paralelos. O objeto da especificação é o módulo SUCO, representando uma entidade, e interagindo com AMBIENTE. Este é composto de dois outros módulos:

- o módulo INTX, representando o provedor de serviço para o SUCO. De um ponto de vista mais amplo, ele representa toda porção restante do SDCD. De um ponto de vista mais restrito, corresponde a uma entidade do nivel INTX. . A interação é feita via soquetes envia e recebe que representam as primitivas de serviço;

- o módulo $APLIC_i$, correspondente a uma entidade da aplicação e, para efeito de ilustração, referente à ECL_i, representando os processos usuários locais do serviço do SUCO. A interação é feita através dos soquetes que implementam as primitivas do SUCO.

Na figura, por questões de simplicidade, usou-se um só nome para representar tanto os soquetes como as interconexões.

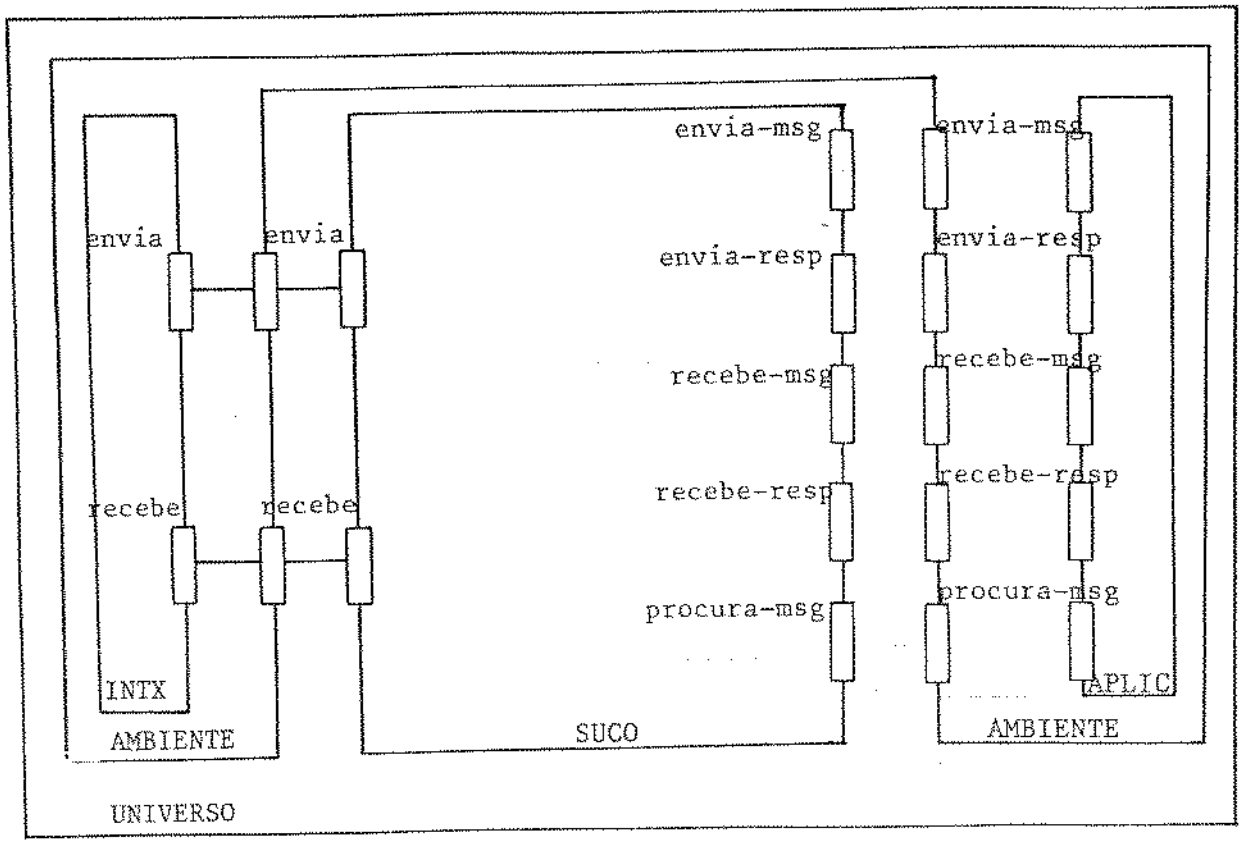


Figura 3.7 Modelo estrutural geral do sistema

A descrição do comportamento dos módulos que compõem AMBIENTE foi feita informalmente na seção 3.4, mas com detalhes suficientes para o prosseguimento da especificação do SUCO. Este, por sua vez, será descrito mais detalhadamente a seguir, até o ponto em que uma implementação num ambiente real seja tarefa fácil.

3.5.3 Modelo Estrutural do SUCO

O módulo SUCO foi particionado em vários módulos, num primeiro passo de refinamento, como mostra a figura 3.8. Alguns deles são módulos terminais e terão seus comportamentos descritos adiante. Outros serão ainda decompostos. A seguir tem-se uma descrição sucinta da função de cada um deles.

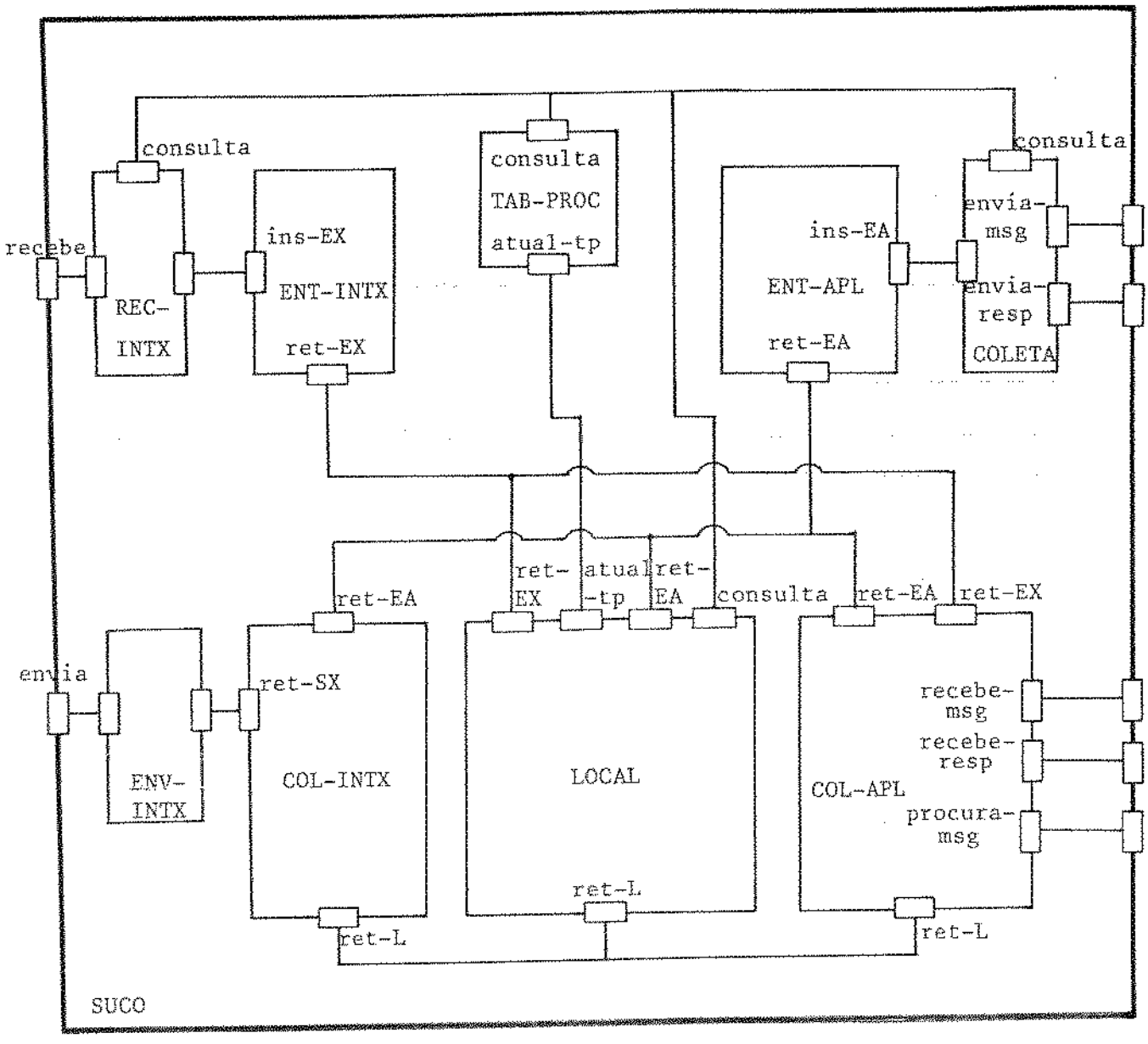


Figura 3.8 Modelo estrutural do SUCO

módulo	descrição
REC-INTX	interface entre SUCO e INTX, para recepção de mensagens. Comanda a execução da primitiva recebe. Insere as mensagens recebidas em uma fila de entrada.
ENV-INTX	interface entre SUCO e INTX, para envio

	de mensagens. Comanda a execução da primitiva envia. Retira as mensagens de uma fila de saída.
COLETA	coletor de mensagens oriundas da aplicação. Implementa as primitivas envia-mensagem e envia-resposta. Será ainda decomposto.
ENT-INTX	fila de mensagens oriundas de INTX.
ENT-APL	fila de mensagens oriundas de APLIC.
TAB-PROC	tabela de identificação de processos endereçáveis. Contém nome, localização física e estado de cada processo.
COL-INTX	coleta e armazena mensagens destinadas a INTX. Será ainda decomposto.
LOCAL	processa mensagens destinadas ao próprio SUCO, gera mensagens e mantém uma fila de saída para mensagens geradas internamente. Será ainda decomposto.
COL-APL	entrega mensagens para a aplicação. Implementa as filas de mensagens destinadas aos processos usuários, bem como as primitivas recebe-mensagem, recebe-resposta e procura-mensagem. Será ainda decomposto.

3.5.4 Comportamento e Refinamento dos módulos do SUCO

O comportamento dos módulos terminais são descritos a seguir. A descrição do comportamento de um módulo terminal será feita buscando-se não entrar demasiadamente em detalhes, a fim de não se inibir iniciativas de implementação e nem alongar excessivamente esta descrição. Desse modo, haverão funções descritas por pseudo-comandos. Antes porém são definidos alguns tipos de dados que serão utilizados nas descrições de comportamento. Na figura 3.9 tem-se um esquema ilustrativo mostrando descritores e formatos de mensagens.

Tipos de dados pré-definidos

```
camada     =  (aplic, suco, intx)
result     =  (ok, nao-achou,buf-cheio, buf-vazio, time-out, erro)
estado-processo = (ativo, inativo)
cop-tp     =  (inclui, exclui, altera, pesq-num, pesq-nome)

           ********************************

corpo      =  array |1..maxcorp| of 0..255

cabecalho =  record of
               tipo-msg   : char;
               destino,
               origem     : identifier;
               classe     : integer;
               contr      : char;
             end


prefixo    =  record of
               flag       : char;
               dest,orig  : integer;
               compr      : integer;
               contr      : char;
             end

sufixo     =  record of
```

```
        CRC, flag : char
      end
```

****

formato de mensagens por nivel

```
ud-aplic  =  corpo              -  formato das mensagens de APLIC

ud-suco   =  record of          -  formato das mensagens de SUCO
               cabec     : cabecalho;
               dado      : ↑ ud-aplic;
             end

ud-intx   =  record of          -  formato das mensagens de INTX
               prefix    : prefixo;
               dado      : ↑ ud-suco;
               sufix     : sufixo;
             end


desc-msg  =  record of          -  descritor de mensagens
                                   internas ao SUCO
               dado      :  ud-suco;
               compr     :  integer;
               no        :  integer;
             end

elem-tp   =  record of          -  entrada tabela identificacao de
                                   processos
               nome      :  identifier;
               num       :  integer;
               no        :  integer;
               nivel     :  camada;
               estado    :  estado-processo;
             end
```

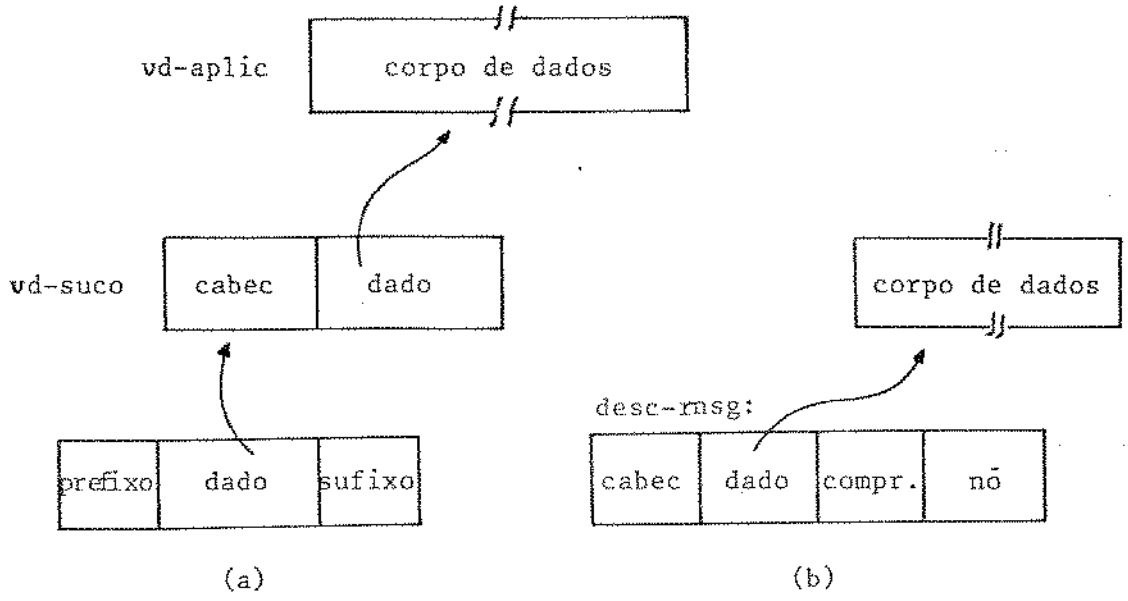
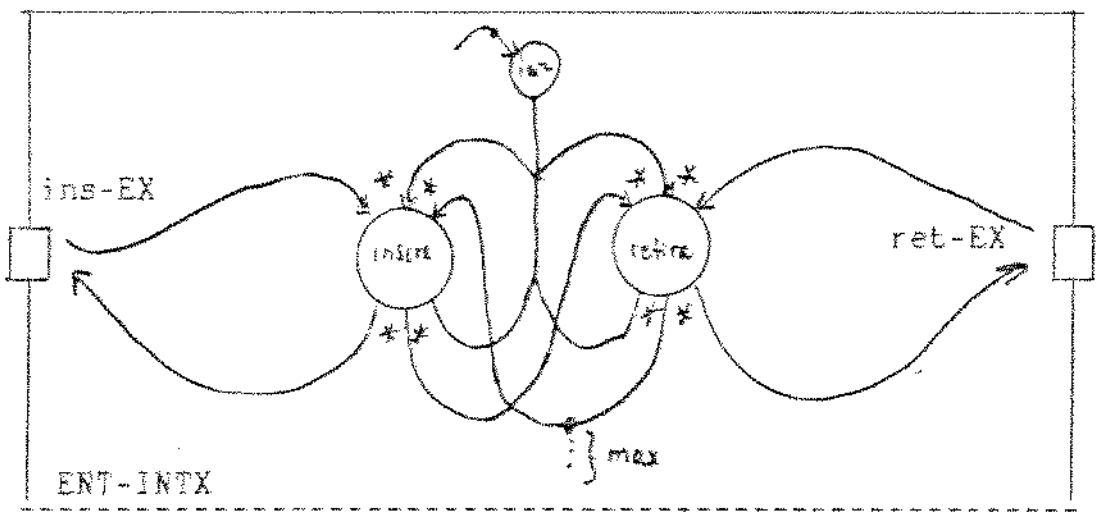


Figura 3.9 (a) Formatos de mensagens por nível
(b) Descritor de mensagens no nível SUCO.

3.5.4.1 Módulo ENT-INTX

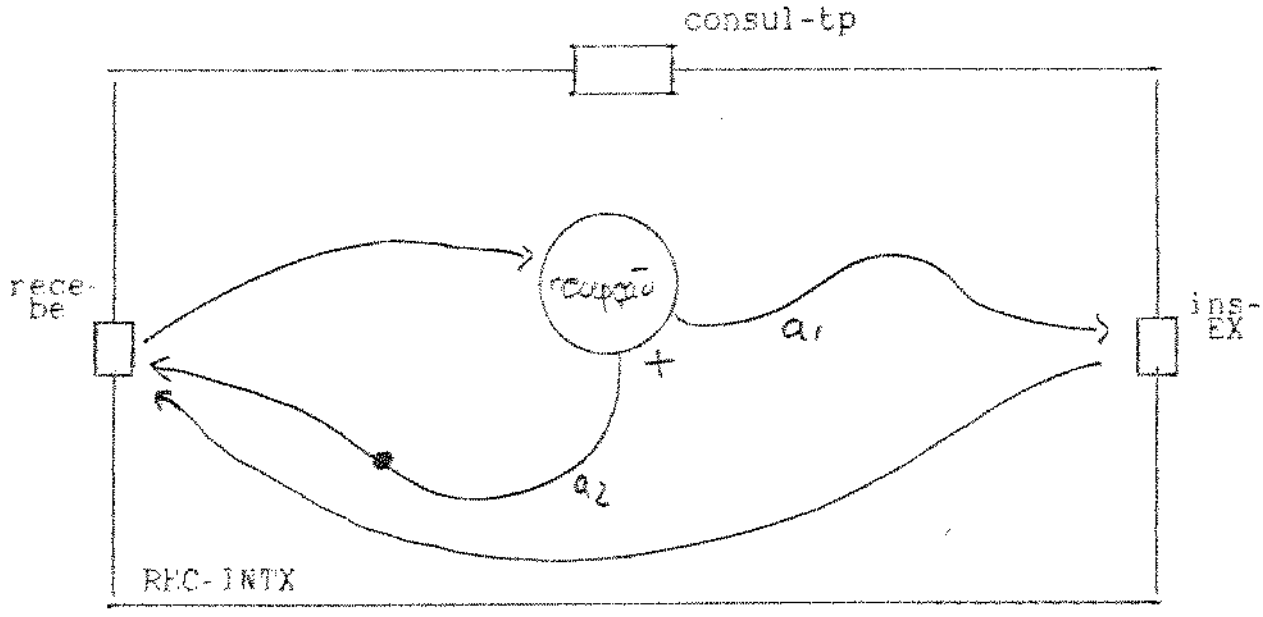



```
module ENT-INTX

  sockets

        ins-EX     : entry-point procedure reentrant
                     parameters  x    : desc-msg  in
                                 nivel : camada in
                     end-parameters

        ret-EX     : entry-point procedure reentrant
                     parameters  x    : desc-msg  out
                                 nivel : camada in
                                 sucesso : result out
                     end-parameters

  end sockets

  data structures

    max = " tamanho maximo da fila"
    " outras estruturas especificas da fila"

  end data structures
```

```
  algorithms

    node insere
      " insere parâmetros recebidos na fila"
    end

    node retira

      " busca elemento segundo parâmetro nivel"
      if "achou elemento" then sucesso := ok
                          else sucesso := nao-achou
    end

    node inic
      " inicializa estrutura de fila"
    end

  end algorithms

end module
```

Obs.: nao é objetivo desta especificacao a implementacao de estruturas particulares de filas.

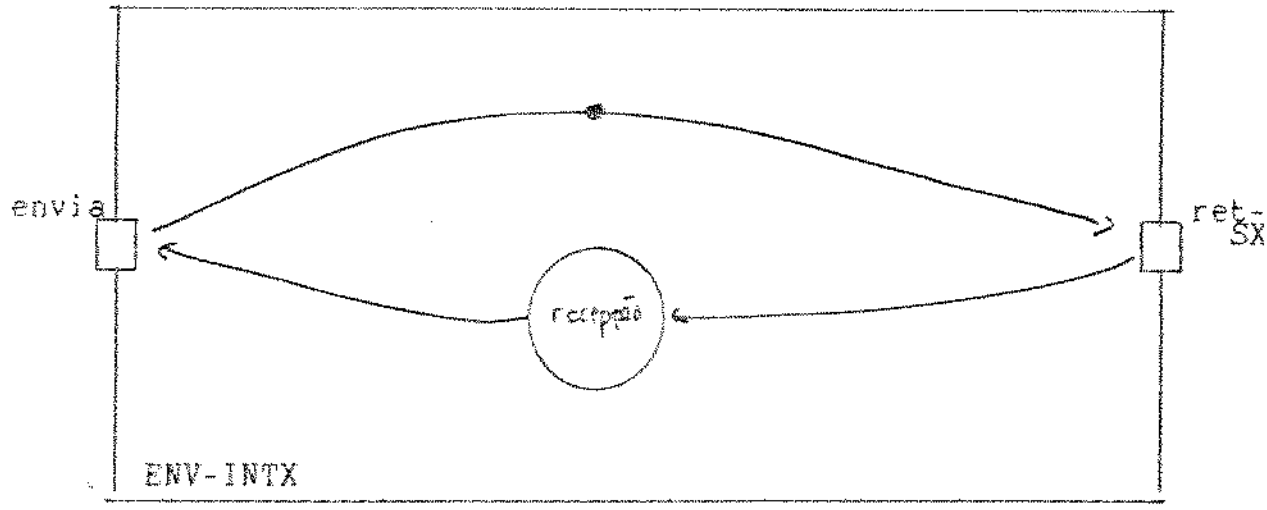
3.5.4.2 Módulo REC-INTX




```
module REC-INTX

  sockets

    recebe     : entry-point procedure reentrant external
                 parameters  no-origem   :  integer in
                             comprimento:  integer in
                             mensagem    :  ud-suco in
                 end-parameters

    ins-EX     : entry-point procedure reentrant external
                 parameters  x   :  desc-msg  out
                             nivel-dest : camada out
                 end-parameters

    consul-tp  : entry-point procedure reentrant external
                 parameters  cod-oner : con-tp out
                             y : elem-tp  in out
                             res : result in
                 end-parameters

  end sockets
```

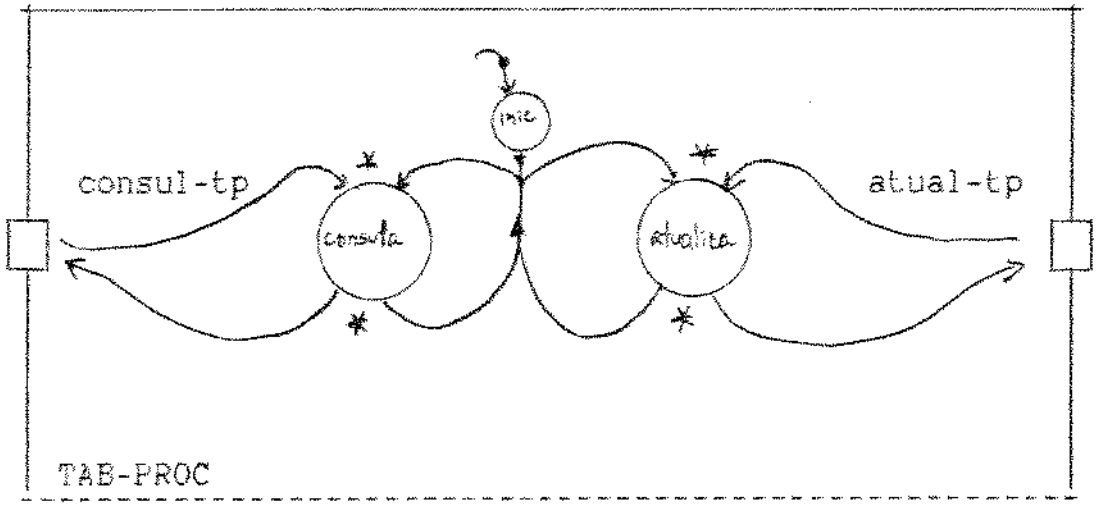
```
algorithms

  node recepcao

    x.dado  := mensagem
    x.compr := comprimento
    x.no    := no-origem

    y.nome  := x.dado.cabec.destino
    consul-tp (pesq-nome,y,res)
    if (y.nivel <> intx) and (estado = ativo)
       then begin
              nivel-dest:= y.nivel; arc a1
            end
       else begin
              " descarta mensagem e libera buffer";
              arc a2
            end
  end node

end algorithms

end module
```

3.5.4.3 Módulo ENV-INTX




```
module ENV-INTX

  sockets

    envia : entry-point procedure reentrant external
                parameters  no-origem   :  integer out
                            comprimento:  integer out
                            mensagem    :  ud-suco out
                end-parameters

    ret-SX    : entry-point procedure reentrant external
                parameters  x   :  desc-msg  in end-parameters

  end sockets

  algorithms

    node recepcao
      mensagem    := x.dado;
      comprimento := x.compr;
      no-destino  := x.no;
    end node

  end algorithms

end module
```

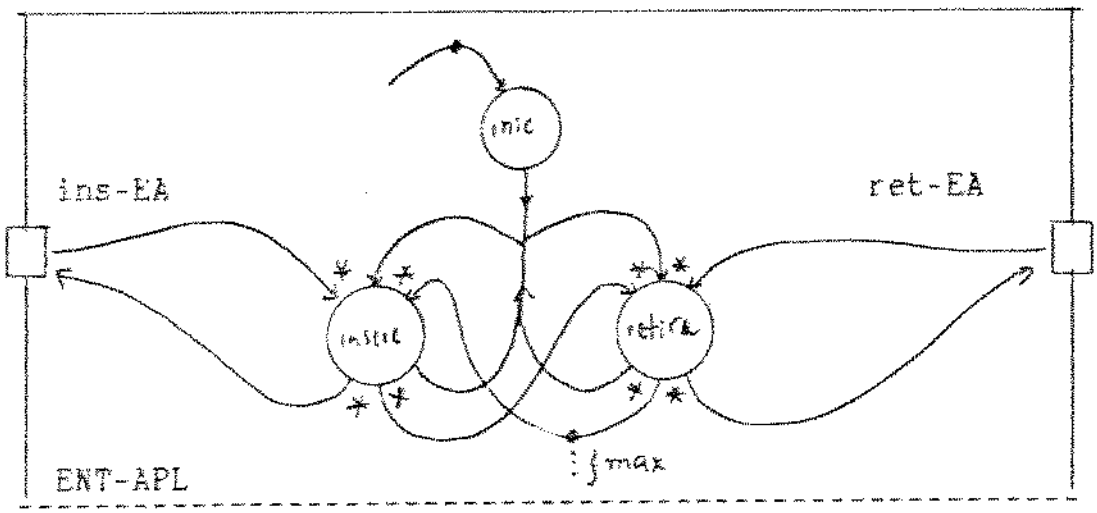
3.5.4.4 Módulo TAB-PROC




```
module TAB-PROC

  sockets

    consul-tp : entry-point procedure reentrant
                parameters  xcop : cop-tp in
                            x : elem-tp in out
                            res-con : result out
                end-parameters

    atual-tp  : entry-point procedure reentrant
                parameters  ycop : cop-tp in
                            y : elem-tp in out
                            res-atl : result out
                end-parameters

  data stuctures

    "estruturas de dados especificas da tabela"

  end data structures
```

```
  algorithms

    node consul-tp

      case xcop of
        pesq-num : "pesquisa tabela, retorna estado do processo"
        pesq-nome: "pesquisa tabela, retorna nó,
                          número e estado"
      end
    end node

    node atualiza

      case ycop of
        inclui: "inclui novo elemento"
        exclui: "exclui elemento"
        altera: "modifica estado de processo"
      end
    end node

    node inic

      "inicializa tabela de processos"

    end node

  end algorithms

end module
```

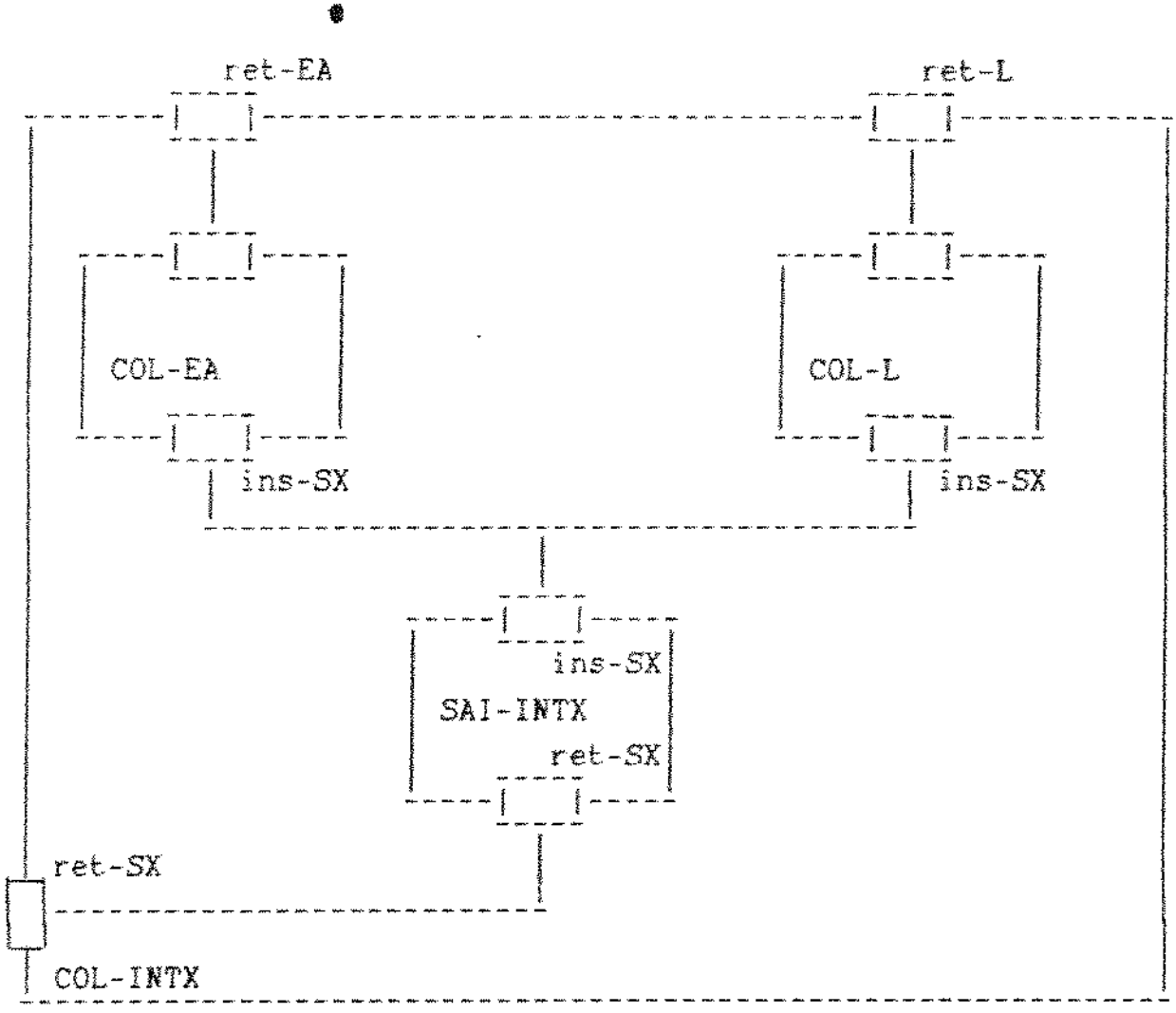
3.5.4.5 Módulo ENT-APL



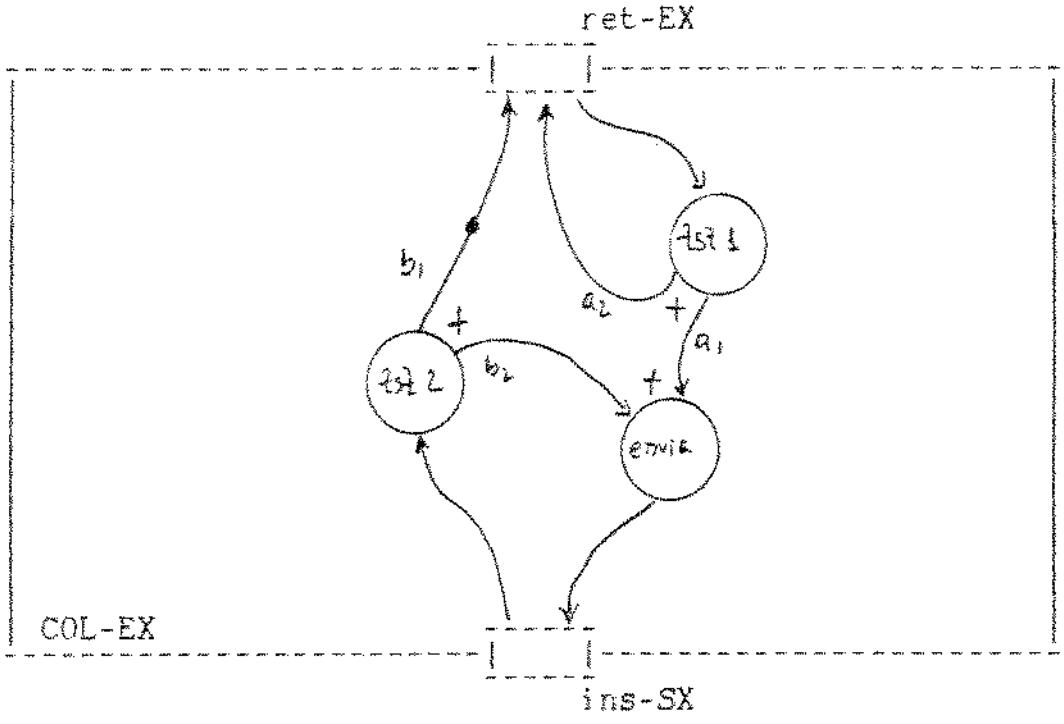

```
module ENT-APL

  sockets

    ins-EA     : entry-point procedure reentrant
                 parameters  x    : desc-msg in
                             nivel : camada in
                 end-parameters

    ret-EA     : entry-point procedure reentrant
                 parameters  x    : desc-msg out
                             nivel : camada in
                             sucesso : result out
                 end-parameters

  end sockets

  data structures

    max = " tamanho maximo da fila"
    " outras estruturas especificas da fila"

  end data structures
```

```
algorithms

  node insere

    " insere parâmetros recebidos na fila"

  end

  node retira

    " busca elemento segundo parâmetro nivel"
    if "achou elemento" then sucesso := ok
                        else sucesso := nao-achou

  end

  node inic

    " inicializa estrutura de fila"

  end

end algorithms

end module
```

3.5.4.6 Refinamento do Módulo COL-INTX



3.5.4.7 Módulo COL-EX



module COL-EX

sockets

ret-EX : entry-point procedure reentrant external
parameters x : desc-msg in
nivel : camada out initial intx
sucl : result in
end-parameters

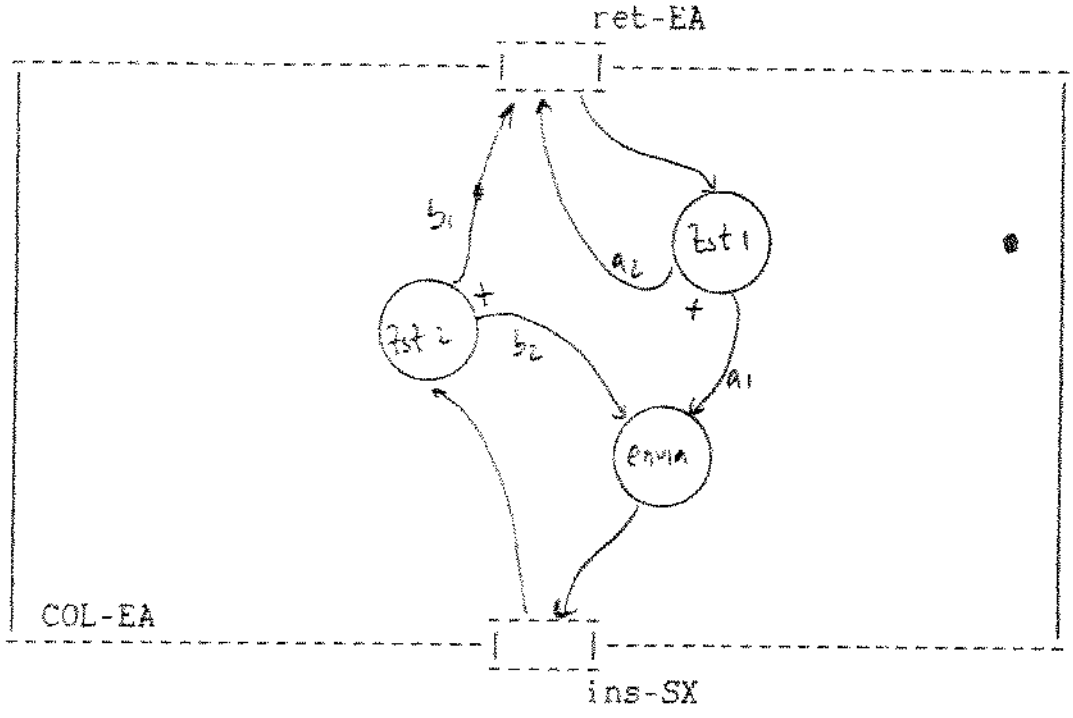
ins-SX : entry-point procedure reentrant external
parameters y : desc-msg out
suc2 : result in
end-parameters

end sockets

```
  algorithms

    node envia

      y := x

    end

    node tst2

      if suc2 = ok
           then arc b1
           else begin wait(1); arc b2 end
    end

    node tst1

      if suc1 = ok
            then arc a1
            else begin wait(1);  arc a2  end
    end

  end algorithms

end module
```

obs.: a primitiva wait("intervalo de tempo") faz com que o processo emitente fique suspenso até decorrer o intervalo especificado.

3.5.4.8 Módulo COL-EA



module COL-EA

sockets

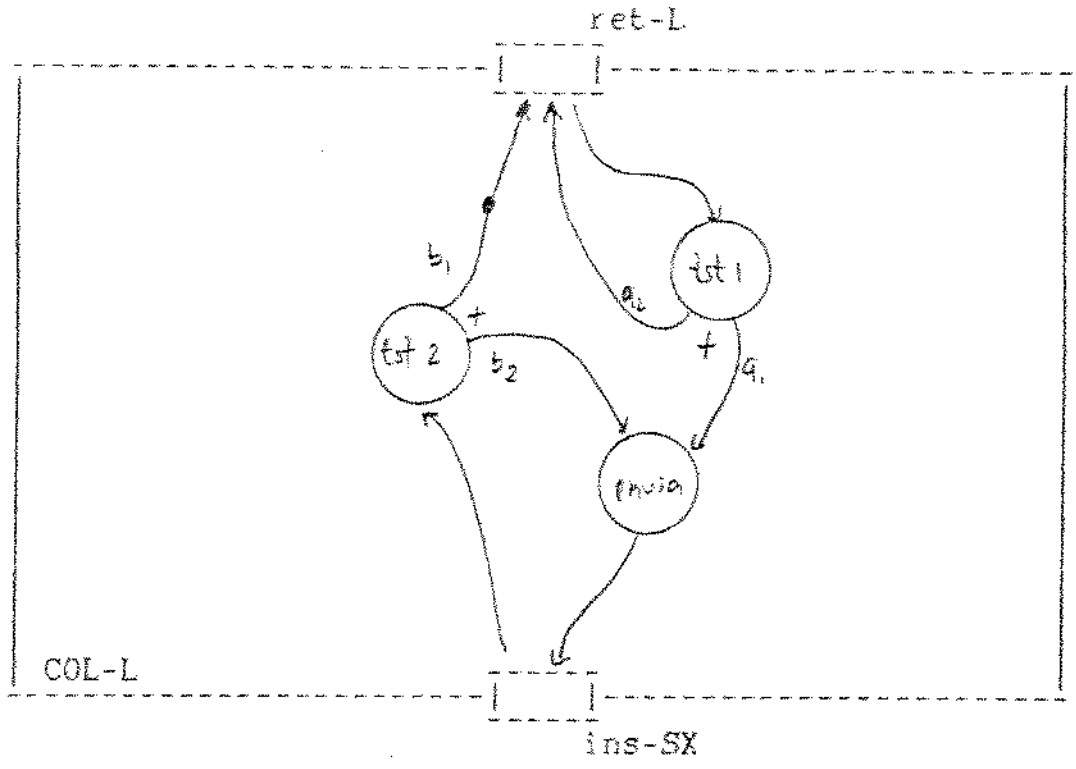
ret-EA : entry-point procedure reentrant external
parameters x : desc-msg in
nivel : camada out initial intx
sucl : result in
end-parameters

ins-SX : entry-point procedure reentrant external
parameters y : desc-msg out
suc2 : result in
end-parameters

end sockets

```
  algorithms

    node envia

      y := x

    end

    node tst2

      if suc2 = ok
           then arc b1
           else begin wait(1); arc b2 end
    end

    node tst1

      if suc1 = ok
            then arc a1
            else begin wait(1);  arc a2  end

    end

  end algorithms

end module
```

3.5.4.9 Módulo COL-L



module COL-L

sockets

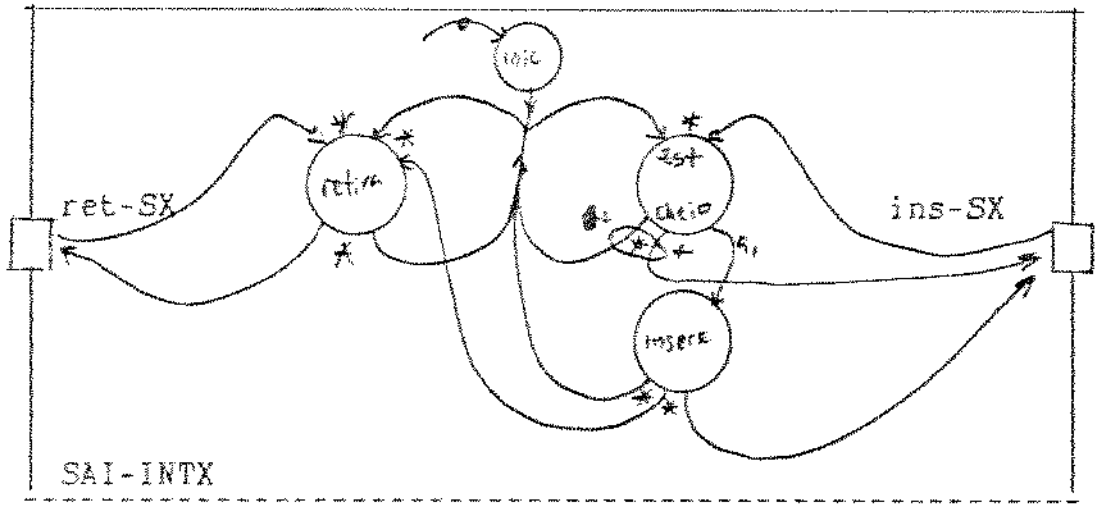
ret-L : entry-point procedure reentrant external
parameters x : desc-msg in
nivel : camada out initial intx
sucl : result in
end-parameters

ins-SX : entry-point procedure reentrant external
parameters y : desc-msg out
suc2 : result in
end-parameters

end sockets

```
algorithms

  node envia

    y := x

  end

  node tst2

    if suc2 = ok
         then arc b1
         else begin wait(1); arc b2 end
  end

  node tst1

    if suc1 = ok
          then arc a1
          else begin wait(1);  arc a2  end
  end

end algorithms

end module
```

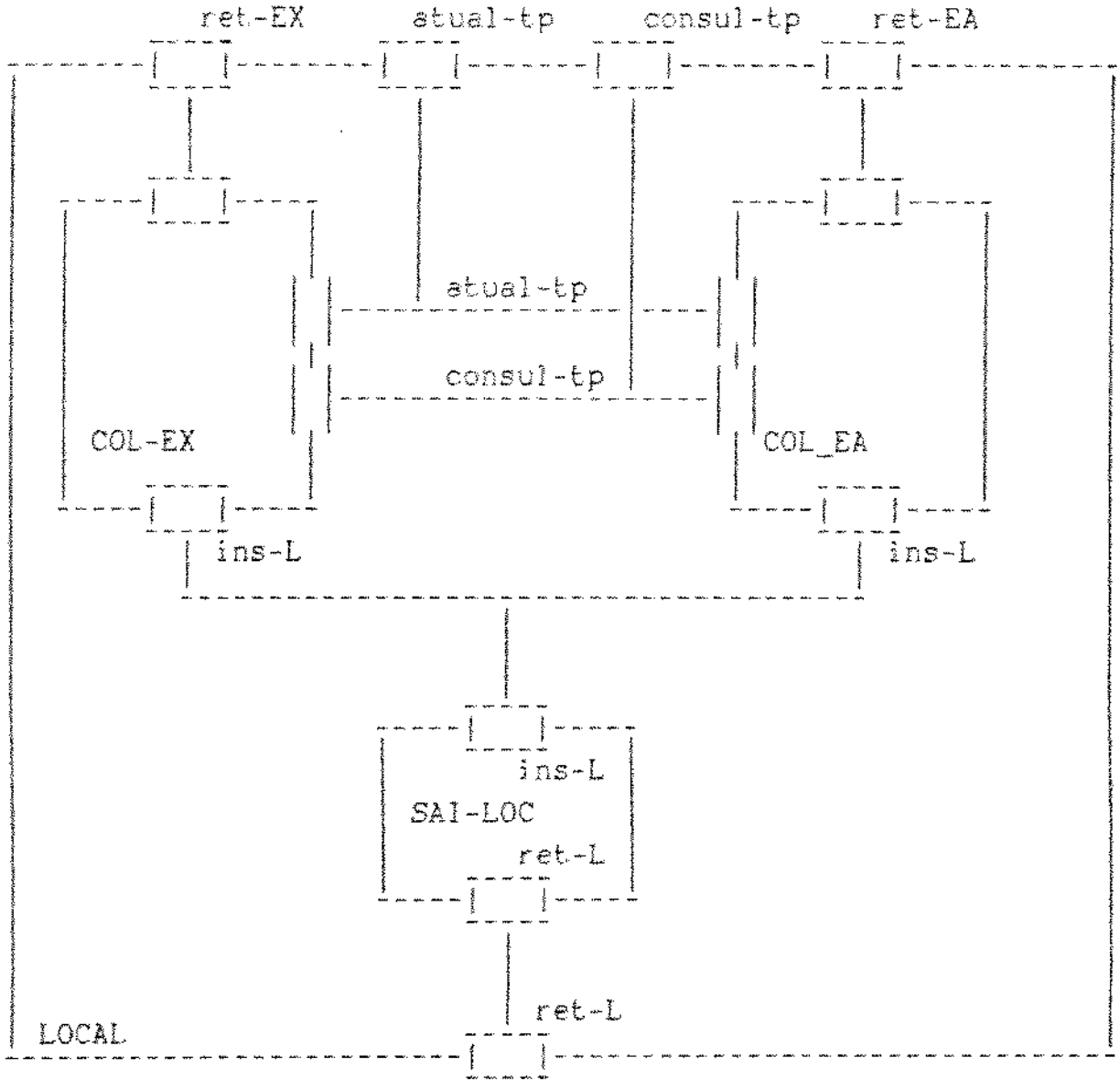
3.5.4.10 Módulo SAI-INTX



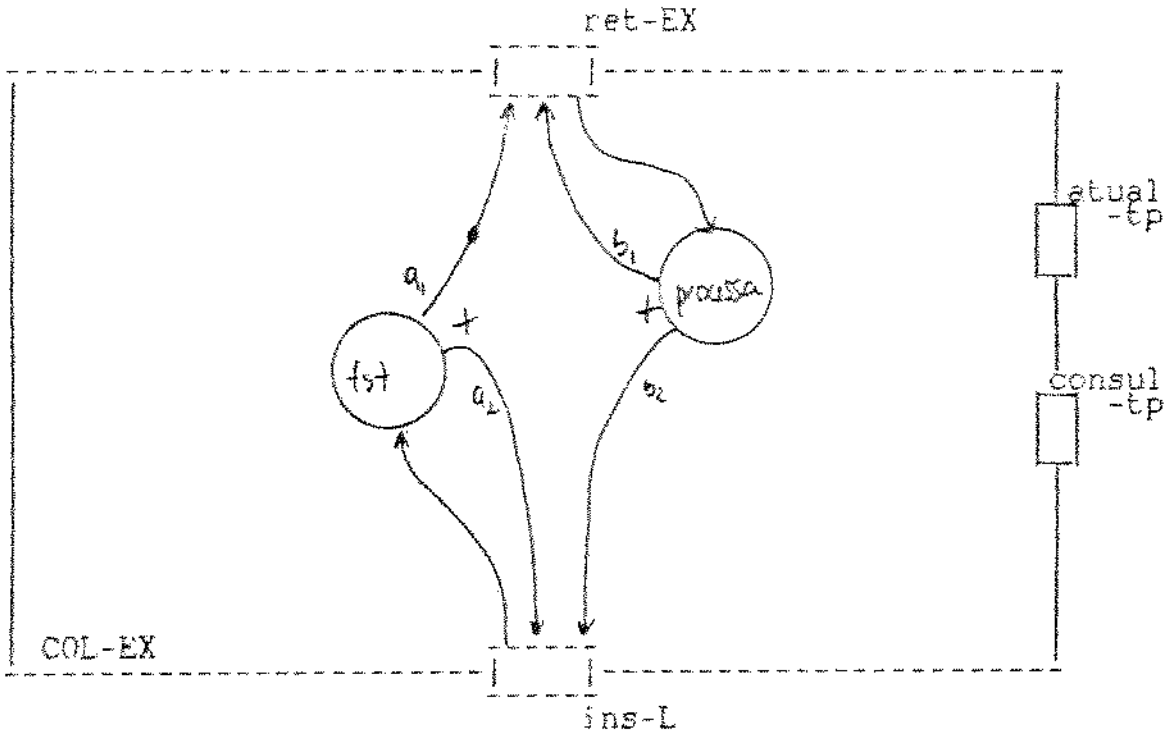

```
module SAI-INTX

  sockets

    ret-SX     : entry-point procedure reentrant
                 parameters  x : desc-msg  out  end

    ins-SX     : entry-point procedure reentrant
                 parameters  y   :  desc-msg  in
                             sucesso: result out
                 end

  end sockets

  data structures

    max = " tamanho máximo da fila"
    full: boolean
    count: integer
    " outras estruturas particulares da fila"

  end data structures
```

```
algorithms

  node retira

    count := count - 1
    full  := false
    "retira mensagem, atualiza ponteiros"

  end

  node tst-cheio

    if full
       then begin sucesso:= buf-cheio; arc a2 end
       else arc a1

  end

  node insere

    "insere mensagem";
    count := count + 1;
    full := count >= max;
    sucesso := ok

  end

  node inic

    full := false;
    count := 0;
    "outras inicializacoes"

  end

 end algorithms

end module
```

3.5.4.11 Refinamento do Módulo LOCAL



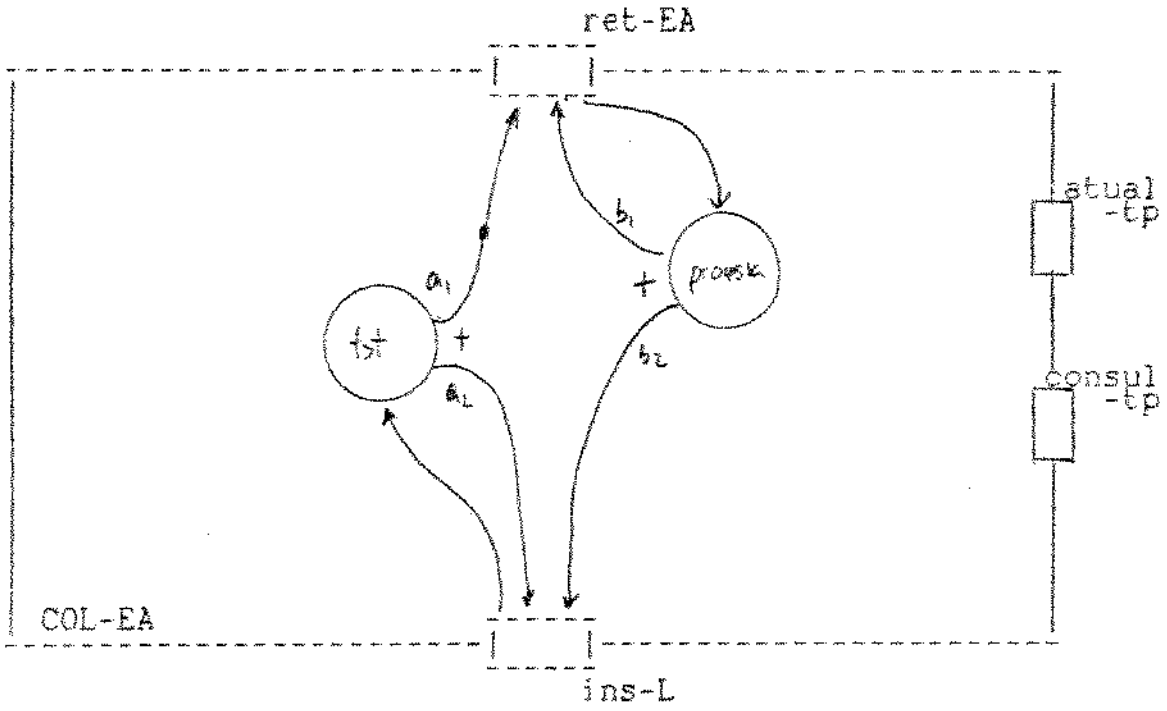
3.5.4.12 Módulo COL-EX




```
module COL-EX

  sockets
    ret-EX     : entry-point procedure reentrant external
                 parameters  x   : desc-msg  in
                             nivel : camada out initial suco
                             suc-ret : result in
                 end-parameters
    ins-L      : entry-point procedure reentrant external
                 parameters  y   : desc-msg  out
                             suc-ins  : result in
                 end-parameters
    atual-tp   : entry-point procedure reentrant external
                 parameters  a-cop : cop-tp out
                             a-tp  : elem-tp out
                             a-res : result in
                 end-parameters
    consul-tp  : entry-point procedure reentrant external
                 parameters  c-cop : cop-tp out
                             c-tp  : elem-tp in out
                             c-res : result in
                 end-parameters
  end sockets
```

```
algorithms

  node processa

    if suc-ret = ok
      then begin
             "consome mensagem, eventualmente gera outra,
              consulta e modifica tabela de processos";

             if "gerou outra" then arc b2
                              else arc b1
           end
      else begin  wait(1);  arc b1 end
    end

    node tst

      if suc-ins = ok then arc a1
                      else begin wait(1); arc a2  end
    end

  end algorithms

end module
```

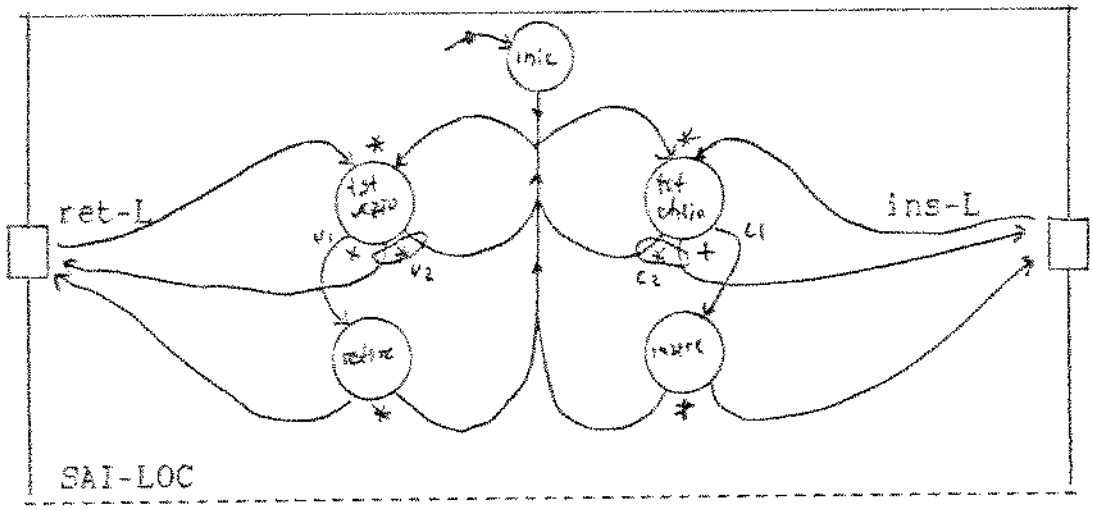
3.5.4.13 Módulo COL-EA




```
module COL-EA

  sockets
    ret-EA    : entry-point procedure reentrant external
                parameters  x    : desc-msg  in
                            nivel : camada out initial suco
                            suc-ret : result in
                end-parameters
    ins-L     : entry-point procedure reentrant external
                parameters  y    : desc-msg  out
                            suc-ins  : result in
                end-parameters
    atual-tp  : entry-point procedure reentrant external
                parameters  a-cop : cop-tp out
                            a-tp  : elem-tp out
                            a-res : result in
                end-parameters
    consul-tp : entry-point procedure reentrant external
                parameters  c-cop : cop-tp out
                            c-tp  : elem-tp in out
                            c-res : result in
                end-parameters
  end sockets
```

```
algorithms

  node processa

    if suc-ret = ok
      then begin
             "consome mensagem, eventualmente gera outra,
             consulta e modifica tabela de processos";

             if "gerou outra" then arc b2
                              else arc b1

           end
      else begin  wait(1);  arc b1 end
    end

    node tst

      if suc-ins = ok then arc a1
                      else begin wait(1); arc a2  end
    end

  end algorithms

end module
```

3.5.4.14 Módulo SAI-LOC



module SAI-LOC

sockets

ret-L : entry-point procedure reentrant
parameters x : desc-msg out
nivel : camada in
suc-ret : result out
end

ins-L : entry-point procedure reentrant
parameters y : desc-msg in
suc-ins : result out
end

end sockets

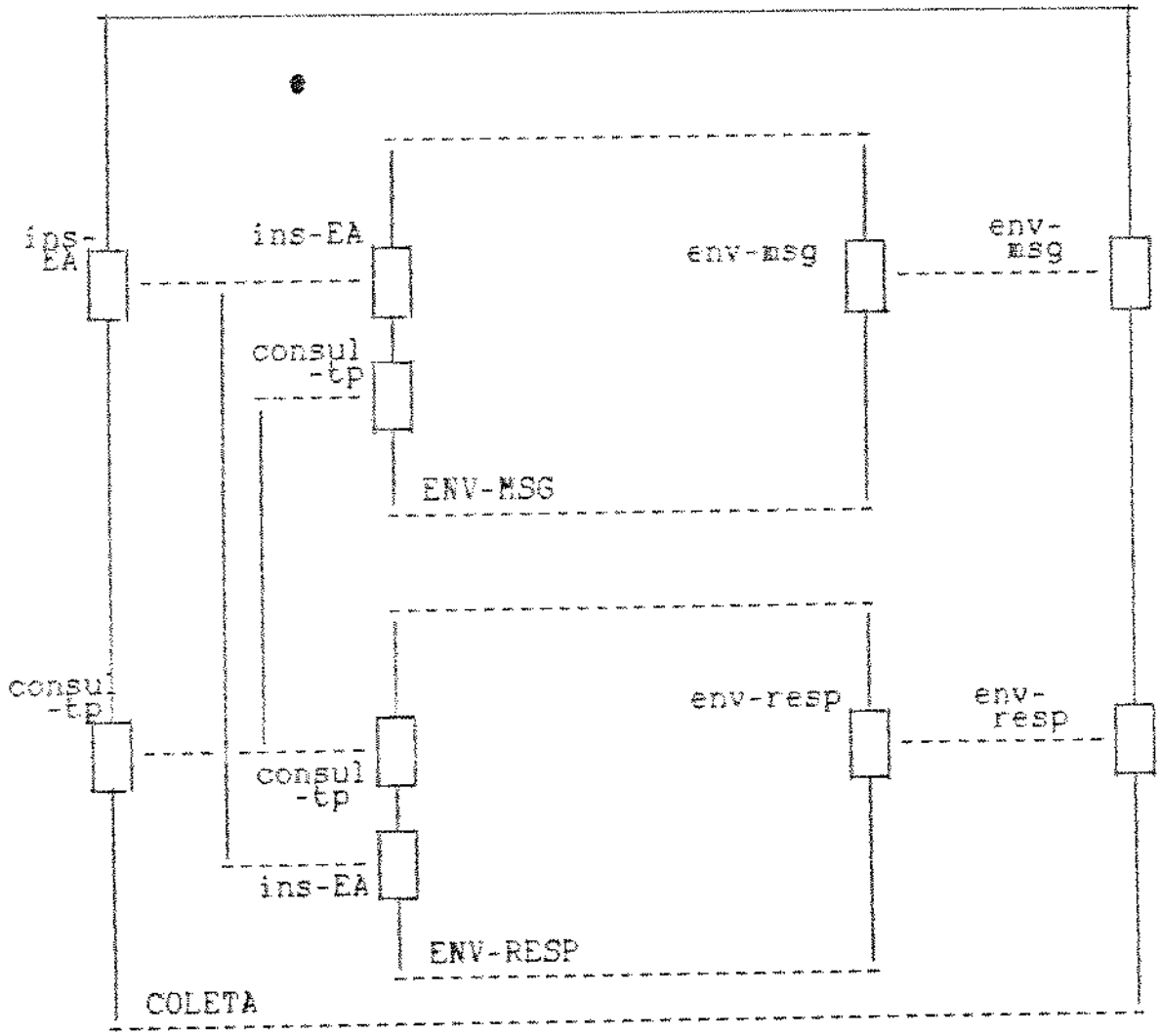
data structures

max = " tamanho máximo da fila"
full, empty: boolean
count: integer
" outras estruturas particulares da fila"

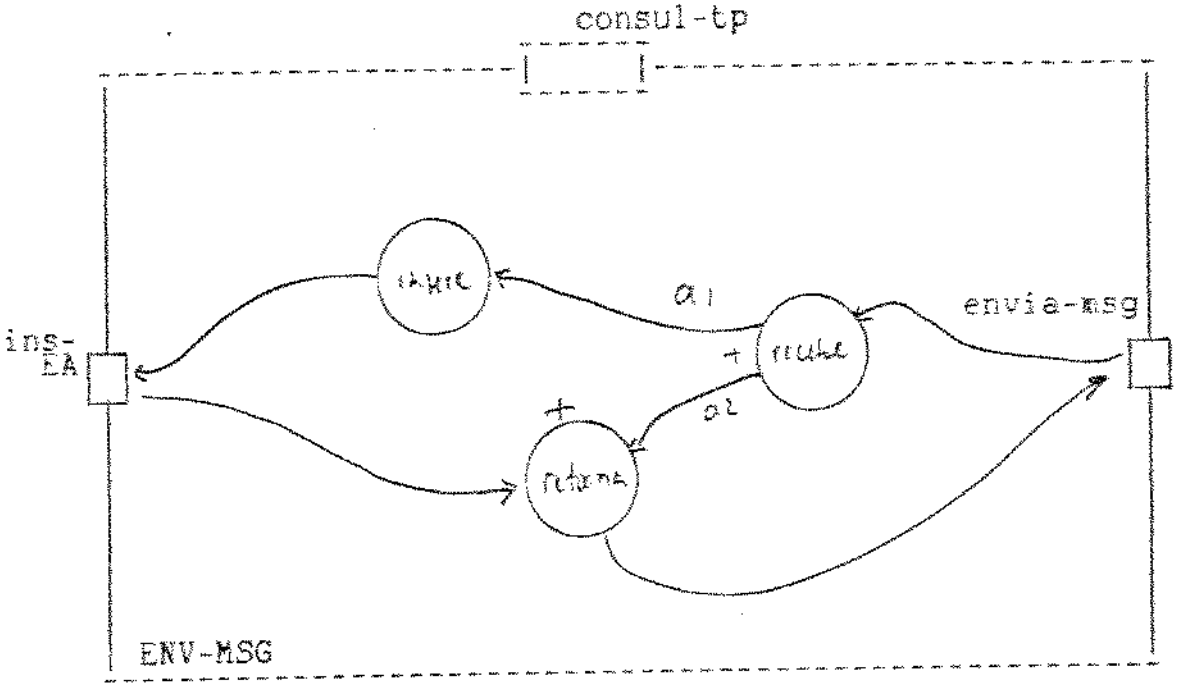
end data structures

```
  algorithms

    node inic

      full := false;  empty := true;   count := 0;
      "outras inicializacoes"

    end

    node tst-vazio
      if empty then begin suc-ret:= buf-vazio; arc v2 end
               else arc v1
    end

    node retira

      "retira mensagem, atualiza ponteiros";
      count := count - 1;
      empty := count = 0;
      full  := false;
      suc-ret := ok

    end

    node tst-cheio

      if full
         then begin suc-ins:= buf-cheio; arc c2 end
         else arc c1

    end

    node insere

      "insere mensagem";
      count := count + 1;
      full := count = max;
      empty := false;
      suc-ins := ok

    end

  end algorithms

end module
```

3.5.4.15 Refinamento do Módulo COLETA



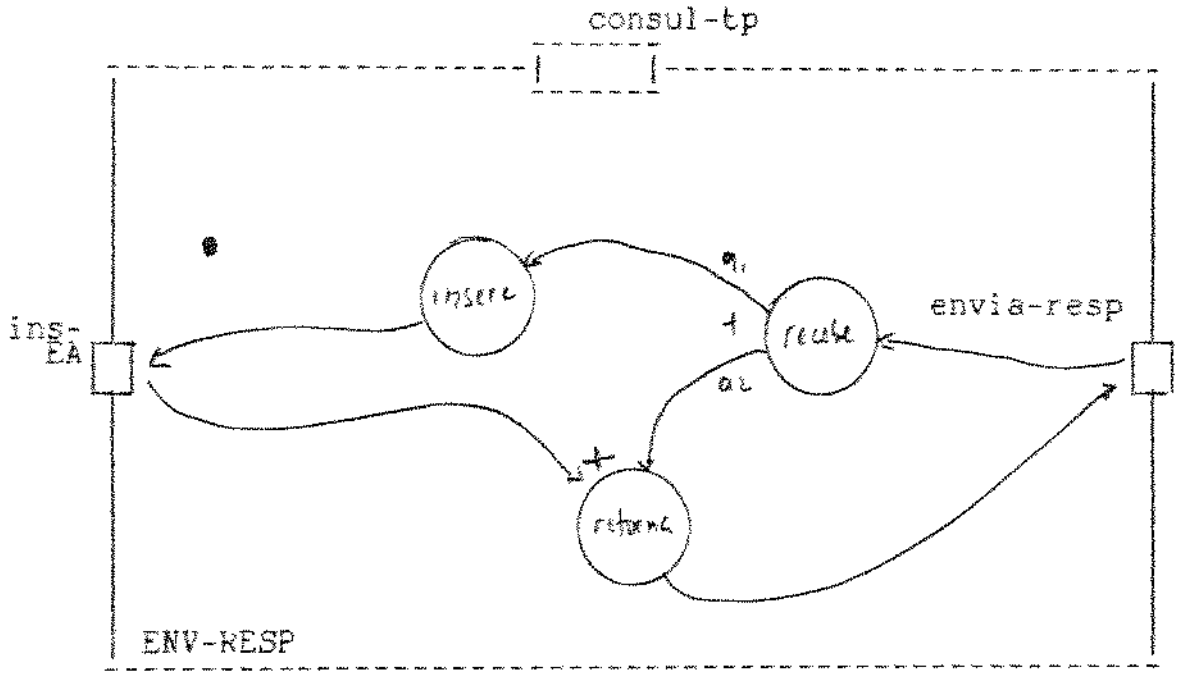
3.5.4.16 Módulo ENV-MSG




```
module ENV-MSG

  sockets

    envia-msg  : entry-point procedure reentrant
                 parameters proc-dest,
                            proc-orig : identifier in
                            compr,
                            classe : integer in
                            mensagem : ^ud-aplic in
                            sucesso : result out
                  end-parameters

    ins-EA     : entry-point procedure reentrant external
                 parameters x : desc-msg out
                            nivel : camada out
                  end-parameters
    consul-tp  : entry-point procedure reentrant external
                 parameters  operacao : cop-tp out
                             y  : elem-tp in out
                             res : result in
                 end-parameters

end sockets
```

```
algorithms

  node recebe
    y.nome := proc-dest;
    consul-tp (pesq-nome, y, res);
    if res = ok then arc a1
                else begin sucesso := erro; arc a2 end
  end

  node insere
    x.dado.cabec.tipo-msg := "codigo para mensagem de dados";
    x.dado.cabec.destino  := proc-dest;
    x.dado.cabec.origem   := proc-orig;
    x.dado.cabec.classe   := classe;
    x.dado.cabec.contr    := "codigo de controle";
    x.dado.dado := mensagem;

    x.compr := compr;
    x.no    := y.no;
    x.nivel := y.nivel;
    sucesso := ok;
  end node
  node retorna end *

 end algorithms

end module
```

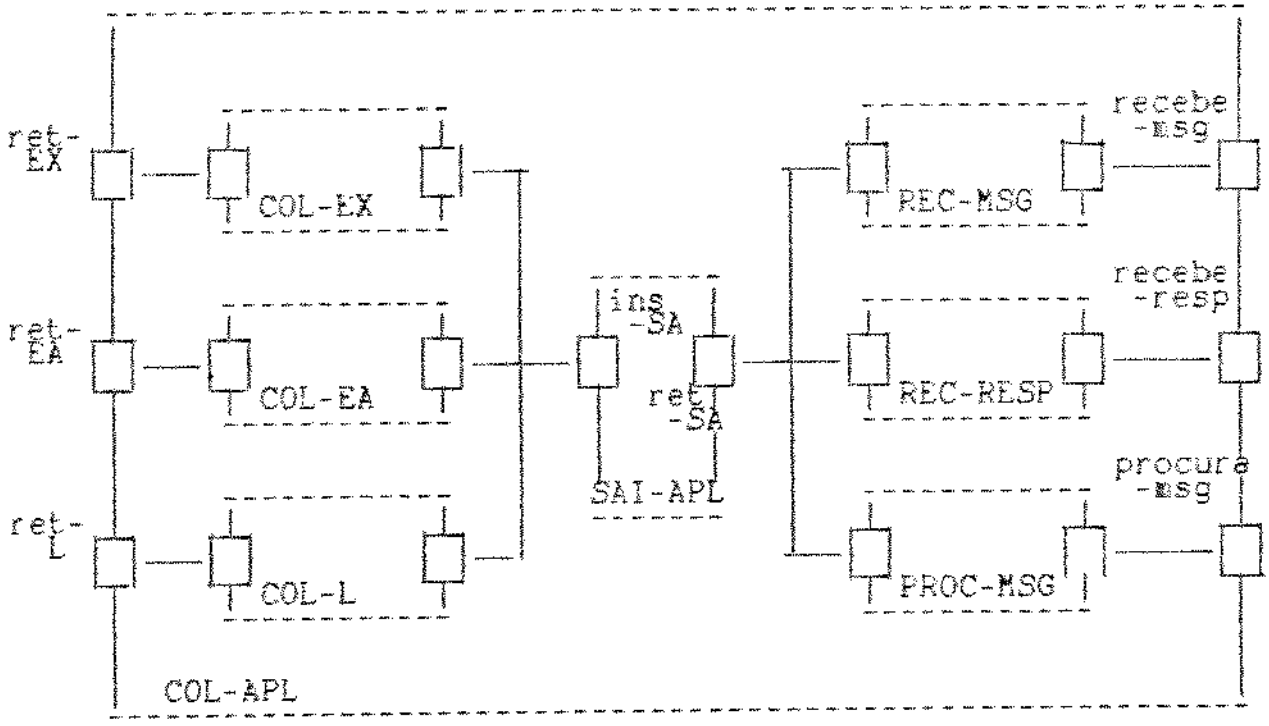
3.5.4.17 Módulo ENV-RESP



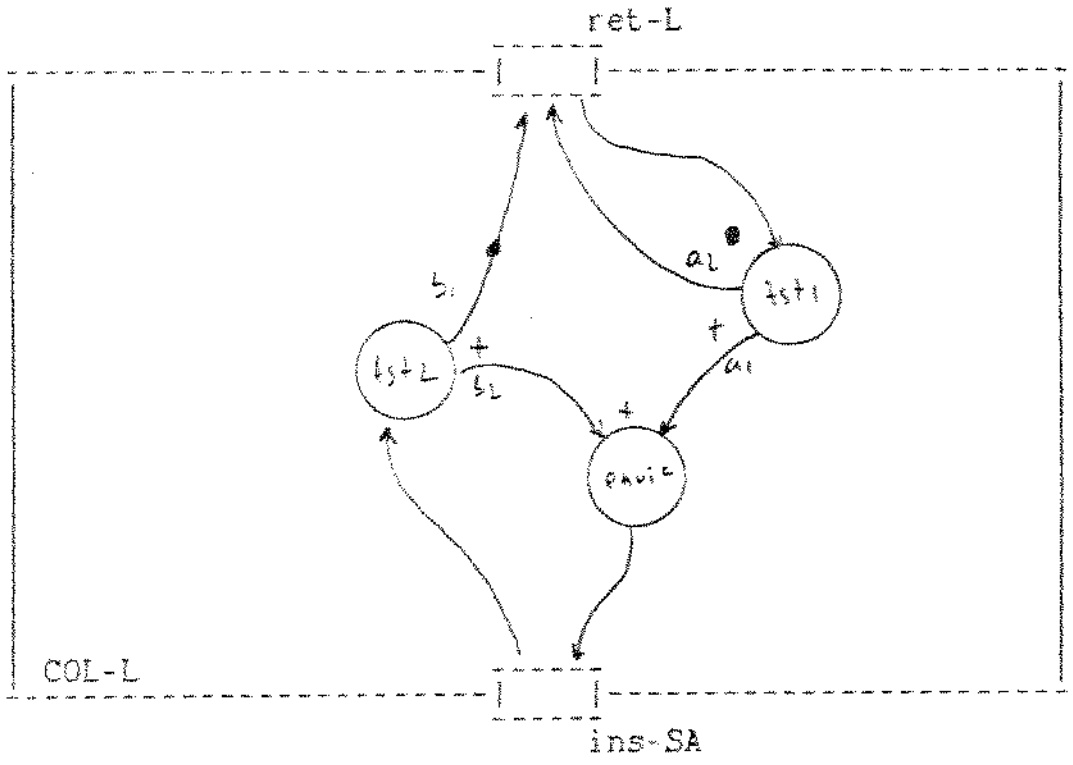

```
module ENV-RESP

  sockets

    envia-msg  : entry-point procedure reentrant
                 parameters proc-dest,
                            proc-orig : identifier in
                            compr,
                            classe : integer in
                            mensagem : ^ud-aplic in
                            sucesso : result out
                  end-parameters

    ins-EA     : entry-point procedure reentrant external
                 parameters x : desc-msg out
                            nivel : camada out
                 end-parameters
    consul-tp  : entry-point procedure reentrant external
                 parameters  operacao : cop-tp out
                             y  : elem-tp in out
                             res : result in
                 end-parameters
end sockets
```

```
algorithms

  node recebe
    y.nome := proc-dest;
    consul-tp (pesq-nome, y, res);
    if res = ok then arc a1
                else begin sucesso := erro; arc a2 end
  end

  node insere
    x.dado.cabec.tipo-msg := "codigo para mensagem de dados";
    x.dado.cabec.destino  := proc-dest;
    x.dado.cabec.origem   := proc-orig;
    x.dado.cabec.classe   := classe;
    x.dado.cabec.contr    := "codigo de controle";
    x.dado.dado := mensagem;

    x.compr := compr;
    x.no    := y.no;
    x.nivel := y.nivel;
    sucesso := ok;
  end node
  node retorna end

end algorithms

end module
```

3.5.4.18 Refinamento do módulo COL-APL



3.5.4.19 Módulo COL-L



module COL-L

sockets

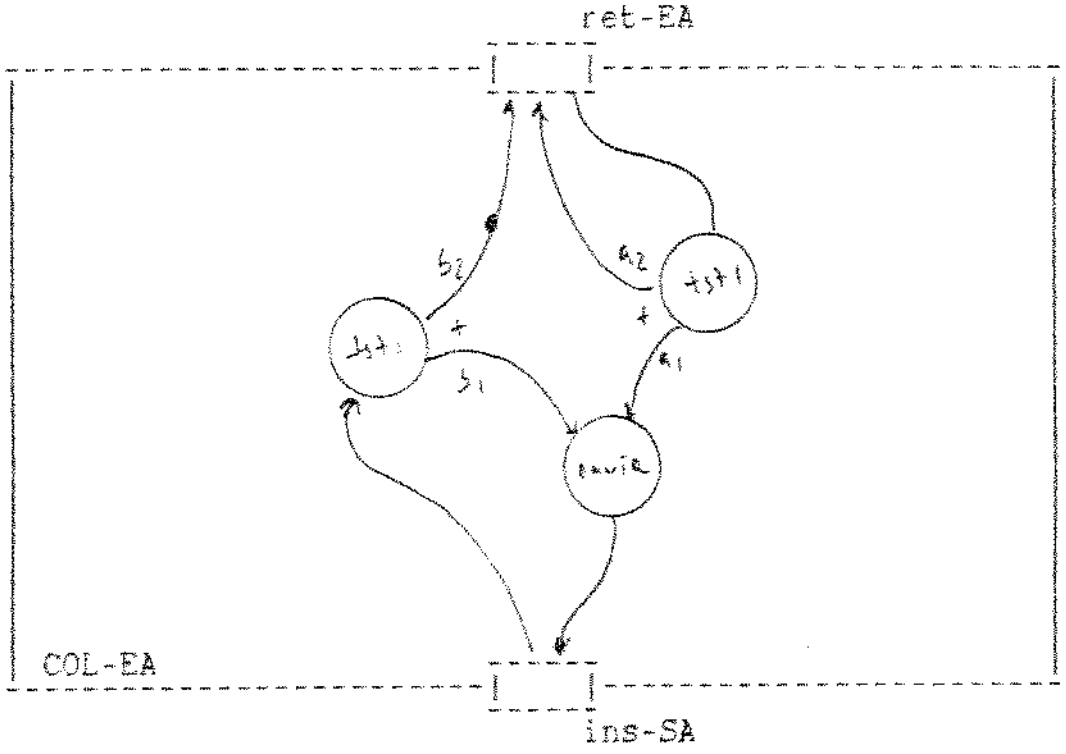
ret-L : entry-point procedure reentrant external
parameters x : desc-msg in
nivel : camada out initial aplic
suc-ret:result in
end-parameters

ins-SA : entry-point procedure reentrant external
parameters y : desc-msg out
suc-ins:result in
end-parameters

end sockets

```
  algorithms

    node envia

      y := x

    end

    node tst2

      if suc-ret = ok
           then arc b1
           else begin wait(1); arc b2 end
    end

    node tst1

      if suc-ins = ok
            then arc a1
            else begin wait(1);  arc a2  end
    end

  end algorithms

end module
```

3.5.4.20 Módulo COL-EA



module COL-EA

sockets

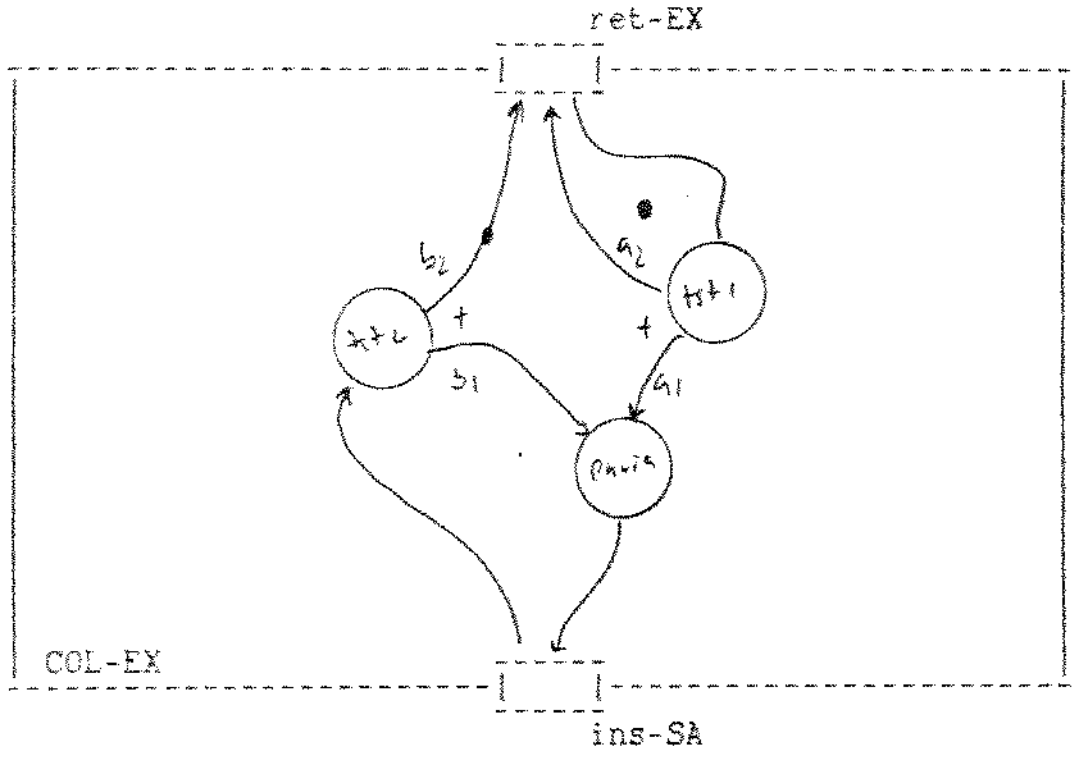
ret-EA : entry-point procedure reentrant external
parameters x : desc-msg in
nivel : camada out initial aplic
suc-ret:result in
end-parameters

ins-SA : entry-point procedure reentrant external
parameters y : desc-msg out
suc-ins:result in
end-parameters

end sockets

```
algorithms

  node envia

    y := x

  end

  node tst2

    if suc-ret = ok
         then arc b1
         else begin wait(1); arc b2 end
  end

  node tst1

    if suc-ins = ok
          then arc a1
          else begin wait(1);  arc a2  end
  end

end algorithms

end module
```

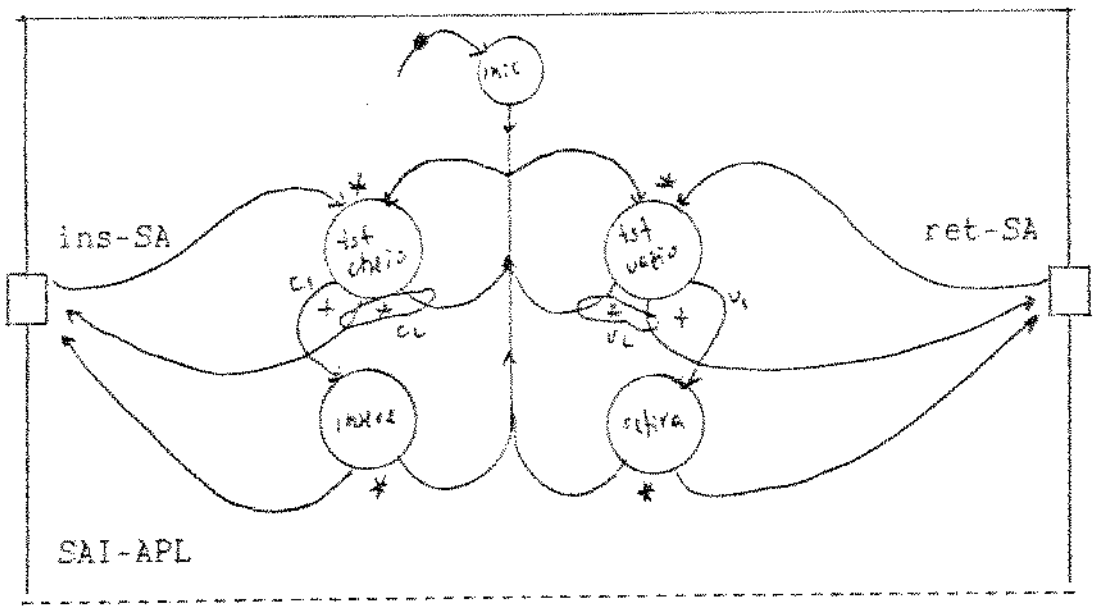
3.5.4.21 Módulo COL-EX




```
module COL-EX

  sockets

    ret-EX     : entry-point procedure reentrant external
                 parameters  x    :  desc-msg  in
                             nivel : camada out initial aplic
                             suc-ret:result in
                 end-parameters

    ins-SA     : entry-point procedure reentrant external
                 parameters  y    :  desc-msg  out
                             suc-ins:result in
                 end-parameters

  end sockets
```

```
  algorithms

    node envia

      y := x

    end

    node tst2

      if suc-ret = ok
           then arc b1
           else begin wait(1); arc b2 end
    end

    node tst1

      if suc-ins = ok
            then arc a1
            else begin wait(1);  arc a2  end
    end

  end algorithms

end module
```

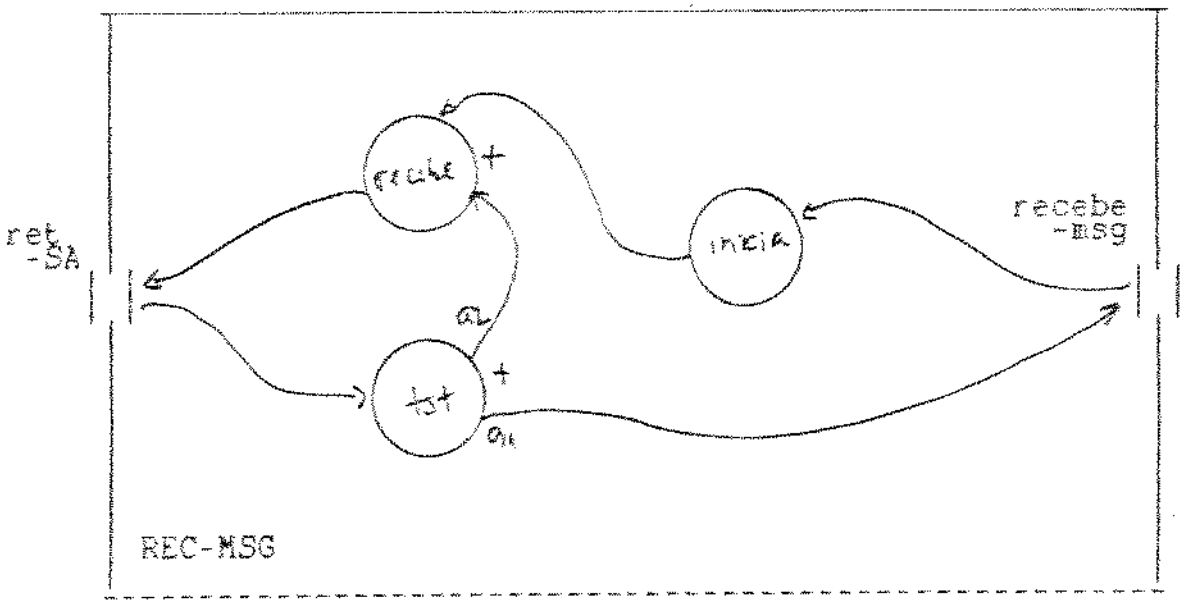
3.5.4.22 Módulo SAI-APL




```
module SAI-APL
  sockets
    ins SA  : entry-point procedure reentrant
              parameters x : desc-msg in
                             suc-ins : result in
              end

    ret-SA  : entry-point procedure reentrant
              parameters
                    y     : desc-msg out
                    destino : identifier in
                    origem  : identifier in
                    orig-especifica : boolean in
                    suc-ret : result out
               end
  end sockets

  data structures

    max = " tamanho máximo da fila"
    full, empty: boolean
    count: integer
    " outras estruturas particulares da fila"
  end data structures
```

```
algorithms

  node inic

    full := false;  empty := true;   count := 0;
    "outras inicializacoes"
  end

  node tst-vazio
    if empty then begin suc-ret:= buf-vazio; arc v2 end
             else arc v1
  end

  node retira

    "retira mensagem, conforme origem pedida";
    if "achou" then begin
                      count := count - 1;
                      empty := count = 0;
                      full  := false;
                      suc-ret := ok

                    end
                else suc-ret := nao-achou
  end

  node tst-cheio

    if full
       then begin suc-ins:= buf-cheio; arc c2 end
       else arc c1
  end

  node insere

    "insere mensagem";
    count := count + 1;
    full := count = max;
    empty := false;
    suc-ins := ok
  end
 end algorithms
end module
```

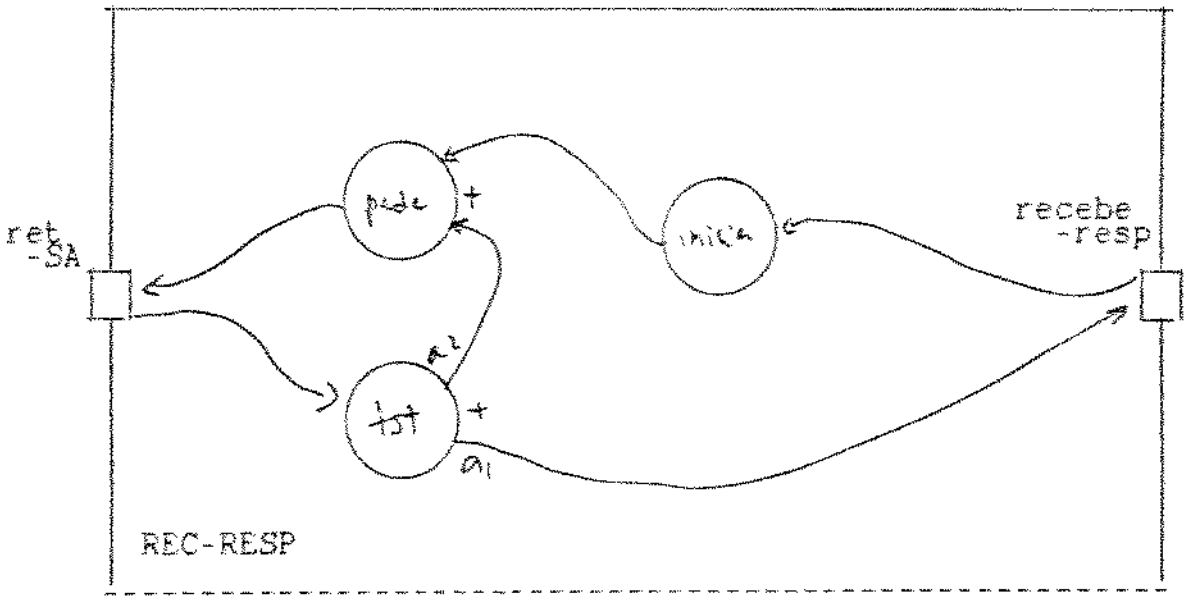
3.5.4.23 Módulo REC-MSG




```
module REC-MSG

  sockets

    recebe-msg : entry-point procedure reentrant
                 parameters
                    proc-orig  : identifier out
                    proc-dest  : identifier in
                    comprimento: integer out
                    classe     : integer out
                    t-espera   : integer in
                    mensagem   : ^ud-aplic out
                    sucesso    : result out
                 end
    ret-SA     : entry-point procedure reentrant external
                 parameters
                    x : desc-msg in
                    destino : identifier out
                    origem  : identifier out
                    orig-especifica : boolean out
                    suc-ret : result in
                 end
end sockets
```

```
algorithms

  node inicia

    orig-especifica := false;
    destino := proc-dest
  end

  node pede  end

  node tst

    if suc-ret = ok
      then begin
             proc-orig    := x.dado.cabec.origem;
             comprimento  := x.compr;
             classe       := x.dado.cabec.classe;
             mensagem     := x.dado.dado;
             sucesso      := ok;
             arc a1
           end
      else if t-espera =< 0
             then begin
                    sucesso := time-out;
                    arc a1
                  end
             else begin
                    wait(1);
                    t-espera := t-espera - 1;
                    arc a2
             end
  end node
end algorithms
end module
```

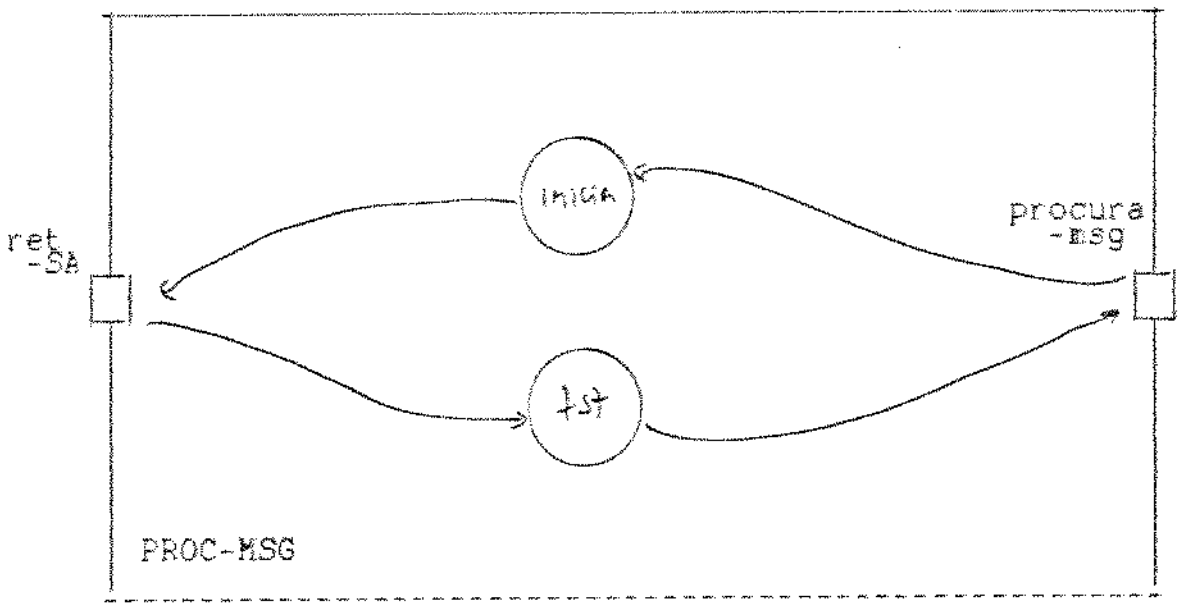
3.5.4.24 Módulo REC-RESP




```
module REC-RESP

  sockets

    recebe-resp: entry-point procedure reentrant
                 parameters
                    proc-orig   : identifier in
                    proc-dest   : identifier in
                    comprimento: integer out
                    classe      : integer out
                    t-espera    : integer in
                    mensagem    : ^ud-aplic out
                    sucesso     : result out
                 end
    ret-SA      : entry-point procedure reentrant external
                 parameters
                    x : desc-msg in
                    destino : identifier out
                    origem  : identifier out
                    orig-especifica : boolean out
                    suc-ret : result in
                end
  end sockets
```

```
algorithms

  node inicia

    orig-especifica := true ;
    destino := proc-dest;
    origem := proc-orig
  end

  node pede  end

  node tst

    if suc-ret = ok
      then begin
             comprimento := x.compr;
             classe      := x.dado.cabec.classe;
             mensagem    := x.dado.dado;
             sucesso     := ok;
             arc a1
           end
      else if t-espera =< 0
             then begin
                    sucesso := time-out;
                    arc a1
                  end
             else begin
                    wait(1);
                    t-espera := t-espera - 1;
                    arc a2
                  end
  end node
 end algorithms
end module
```

3.5.4.25 Módulo PROC-MSG




```
module PROC-MSG

  sockets

    recebe-resp: entry-point procedure reentrant
                 parameters
                    proc-orig   : identifier out
                    proc-dest   : identifier in
                    comprimento: integer out
                    classe      : integer out
                    mensagem    : ^ud-aplic out
                    sucesso     : result out
                 end
    ret-SA      : entry-point procedure reentrant external
                 parameters
                    x : desc-msg in
                    destino : identifier out
                    origem  : identifier out
                    orig-especifica : boolean out
                    suc-ret : result in
                end
  end sockets
```

```
algorithms

  node inicia

    orig-especifica := false;
    destino := proc-dest;
  end

  node tst

    if suc-ret = ok
      then begin
             proc-orig   := x.dado.cabec.origem;
             comprimento := x.compr;
             classe      := x.dado.cabec.classe;
             mensagem    := x.dado.dado;
             sucesso     := ok;
           end
      else sucesso := nao-achou
  end node

end algorithms

end module
```

3.5.5 Experiência de Implementação

A especificação de implementação desenvolvida na seção anterior foi transformada num programa escrito em linguagem de alto nível. Os objetivos dessa programação foram o de verificar a complexidade e de conhecer o porte do programa resultante. Na ausência de uma rede para implementar o programa e executar testes, o trabalho foi desenvolvido em um único computador, no caso um microcomputador. A linguagem escolhida foi o Edison, especificada por Brinch Hansen (1981). E' uma linguagem de programação concorrente, muito simples e própria para desenvolvimento de sistemas.

A estrutura do programa resultante se assemelha à estrutura da especificação de implementação. A sub-rede de comunicação (AMBIENTE.INTX) foi simulada por duas rotinas envia e recebe, chamadas do SUCO. A aplicação (AMBIENTE.APLIC) foi simulada por um conjunto de processos concorrentes que usam as rotinas primitivas do SUCO. O SUCO se compõe de processos concorrentes para executar o processamento sobre mensagens e de estruturas tipo monitor para implementar as estruturas de dados como filas e tabela de endereços de processos.

O programa resultante se apresentou pequeno (da ordem de 700 linhas de código Edison) e sem nenhuma dificuldade maior. As facilidades oferecidas pela linguagem contribuiram para tal.

Esse trabalho deve prosseguir no DCC-UFMG, onde existem projetos de pesquisa e desenvolvimento definidos para redes locais. Os projetos vão desde a definição e implementação de uma sub-rede de comunicação, passando por protocolos de alto nivel, até aplicações de controle de processos.

3.6 CONCLUSOES

Neste capitulo tratou-se de dois aspectos envolvidos na automação industrial com sistemas distribuidos: a aplicação e a comunicação entre processos.

O aspecto de aplicação foi abordado tomando-se como referência as aplicações de SDCD para controle de processos industriais. Inicialmente foram levantadas as caracteristicas, a nivel de comunicação, dessas aplicações. Entre as caracteristicas incluem-se o tipo de informação trocada, o volume, a periodicidade e o formato de mensagens. Num segundo passo foi elaborada uma proposta de um modelo de programas paralelos para servir de ferramenta metodológica de auxilio na concepção de programas de aplicação. O conjunto de programas aplicativos foi designado programa paralelo. Este se constitui de módulos, processos e mensagens. Módulos são compostos de processos que se comunicam por troca de mensagens.

O modelo proposto e as caracteristicas da comunicação geraram alguns requisitos de comunicação entre processos computacionais. O segundo aspecto, da comunicação entre processos, tratou do atendimento destes requisitos. Definiu-se então uma camada na arquitetura do sistema distribuido para dar suporte à comunicação da aplicação - o SUCO. Este implementa os serviços especificados pelas operações primitivas requisitadas pelo nivel de aplicação - comunicação e sincronização entre processos. Ademais, esconde da aplicação os aspectos particulares de implementação da sub-rede de comunicação.

A implementação dos algoritmos da camada SUCO tem duas alternativas principais. A primeira, que foi adotada neste trabalho, consiste na incorporação das rotinas que implementam os serviços como parte dos programas aplicativos. A segunda alternativa consiste na incorporação dessas rotinas a cada sistema operacional local. Vê-se esta alternativa como um passo seguinte de pesquisa na continuidade deste trabalho.

A localização do SUCO, do ponto de vista da padronização ISO, foi em favor de situá-lo na camada de sessão. Todavia, como dito no início do capítulo, não se descarta a discussão em torno da localização de partes do SUCO na camada de transporte. A questão é que o modelo da ISO pretende ser genérico, ao passo que a automação industrial com sistemas distribuídos tem um caráter específico.

A interface do SUCO com a sub-rede de comunicação foi definida para atender aos propósitos deste trabalho, através das primitivas envia e recebe e seus parâmetros. A utilização de serviços de camadas inferiores reais e padronizadas é viável sem maiores implicações no projeto do SUCO. Na especificação de uma camada, o que é definido e padronizado são os serviços, e não o modo de acessá-los. Este depende de cada implementação particular.

4. ANALISE DE DESEMPENHO DE SDCD

4.1 INTRODUÇÃO

O objetivo deste capitulo é proceder à análise de desempenho dos aspectos de comunicação tratados no capitulo 3, feita a partir de modelos matemáticos e de resultados de desempenho obtidos por simulacão destes mesmos modelos. O estudo é sempre feito no âmbito de SDCD, composto de vários subsistemas hierarquizados , um dos quais é o subsistema de suporte à comunicacão da aplicacão (SCOM), que tem como camada exterior o SUCO.

A análise de desempenho, tal como levada neste trabalho, busca encontrar respostas para os seguintes tipos de questões:

. quais devem ser as previsões de memória para armazenamento dos dados que fluem entre os diversos componentes?

. quanto da capacidade de processamento da UCP de um certo elemento de computacão está sendo usada?

. qual o tempo esperado que uma mensagem leva para ir de um ponto a outro do sistema?

A primeira parte deste capitulo trata da simulacão de redes de filas. Inicialmente apresentam-se os conceitos envolvidos e depois descreve-se um simulador desenvolvido especificamente para simular rede de filas. A segunda parte consiste na elaboracão de um modelo de rede de filas representando um SDCD simplificado, porém tipico. Este modelo servirá de base para os estudos subsequentes. O restante do capitulo consiste de estudos de aspectos particulares ou de partes reduzidas do modelo básico, o que se dá efetuando as seguintes etapas: elaboracão de um novo modelo, reduzido ou não; estabelecimento dos valores de variáveis e parâmetros; simulacão e , finalmente, análise dos resultados.

Cumpre ressaltar que o conteúdo deste capitulo, além de servir aos objetivos básicos, de analisar o subsistema de suporte à comunicação da aplicação especificado e seu ambiente, apresenta também ferramentas para auxilio no estudo de sistemas distribuidos, em particular de SDCD, nos aspectos de

especificação,

projeto,

e avaliação.

A tarefa de especificar as características desejadas de um SDCD, para um processo o qual se quer controlar, nem sempre é fácil quando se leva em consideração as restrições de custo, disponibilidade de mercado, adequabilidade de funções, expansibilidade, entre outras. O objetivo é encontrar uma configuração que atenda aos requisitos do processo e às restrições referidas.

Da mesma forma, ao se projetar um SDCD que atenda a uma classe de processos, necessário se faz o bom dimensionamento dos seus componentes, levando em conta várias restrições do tipo visto acima.

O ato de se expandir um SDCD em funcionamento, adicionando componentes ou aumentando as capacidades de cada componente, requer um estudo prévio do desempenho daquele, de maneira a identificar pontos de possiveis alterações. Esta tarefa de avaliação assemelha-se em parte à da especificação.

Neste sentido o conteúdo deste capitulo se apresenta como parte de uma metodologia para concepção e avaliação de sistemas distribuidos para automação industrial. Para isto deve sofrer algumas modificações afim de se tornar um pouco mais geral.

Por não constituir o objetivo central do trabalho, a apresentação das ferramentas é feita de forma sucinta, sem se alongar nos exemplos de simulação e análise.

4.2 SIMULAÇÃO DE REDES DE FILAS

4.2.1. Conceitos Básicos de Redes de Filas

Nesta seção apresenta-se resumidamente o conceito de rede de filas, que é um modelo onde se considera a existência de mais de um sistema de filas operando assincrona e concorrentemente (Kleinrock,1975a-b; Kobayashy,1978; Sauer & Chandy, 1981). A matéria que estuda matematicamente as filas chama-se teoria de filas e é altamente baseado na teoria de probabilidades.

4.2.1.1 Sistemas de filas

A estrutura básica de um sistema de filas considera a existência de uma estação de serviço e de uma fila para elementos que chegam e ali ficam à espera de serem servidos, como mostra a figura 4.1

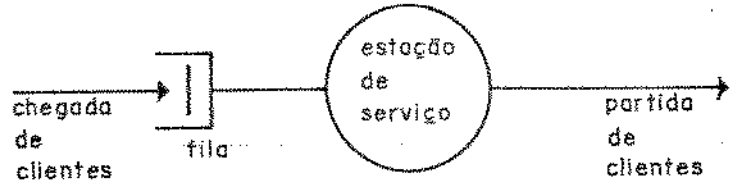


Figura 4.1 - Estrutura básica de um sistema de filas.

Os elementos que fluem através do sistema de filas são chamados genericamente de clientes. Uma sequência de clientes chega à estação de serviço de acordo com um padrão ou processo de chegadas. Se o cliente recém-chegado encontrar a estação ocupada, ele entra na fila de espera. A um certo tempo será selecionado para serviço de acordo com alguma disciplina de escalonamento. O cliente é então servido (usa a estação), após o que deixa o

sistema. Uma estação de serviço pode se constituir de um ou mais servidores, que são os elementos que atendem os clientes, um de cada vez.

4.2.1.2 Caracterização dos sistemas de filas

Os clientes originam-se de uma população ou fonte de entrada. A população tem um tamanho, que pode ser finito ou infinito. Uma caracteristica a ser considerada com relação aos clientes é o padrão estatistico a partir do qual os clientes que chegam são gerados no tempo. O processo de chegadas mais simples é aquele que os clientes chegam em intervalos regulares. Um outro padrão de chegadas, mais simples para tratamento matemático e mais usual em modelamento é o processo de chegadas completamente aleatórias, referido também como processo Poisson de chegadas. Este processo será considerado na maioria das vezes neste trabalho. O inverso do intervalo médio entre chegadas é definido como taxa de chegada de clientes.

O montante de serviço requerido por um cliente é chamado de demanda de serviço, cuja unidade varia dependendo da natureza do servidor e seus clientes. Por exemplo, se o servidor é uma linha de transmissão e os clientes são mensagens, a unidade de serviço será "bit" ou "byte". A demanda de serviço dos clientes dá-se conforme uma distribuição de serviço. A mais simples e prática é a distribuição exponencial. A capacidade de serviço de um servidor identifica quão rapidamente ele atende às requisições de serviço. Se a demanda de serviço é "s" unidades e a capacidade do servidor é "c" unidades/segundo, define-se como tempo de serviço a relação "s/c". Seu inverso é definido como taxa de serviço.

O que caracteriza como os clientes são escalonados para serviço é a disciplina de escalonamento. A regra mais familiar é aquela na qual os clientes são escalonados na mesma ordem da chegada, a disciplina "primeiro-a-chegar,primeiro-a-ser-servido" (FCFS, do inglês "first-come,first-served"). Disciplinas que tratam os clientes de maneira diferenciada são chamadas de "escalonamento

com prioridades", nas quais a ordem de atendimento é baseada nas caracteristicas dos clientes. Existem outras disciplinas que não serão aqui referenciadas.

Na maioria das vezes assume-se que os clientes são homoqêneos, no sentido que têm a mesma distribuicão de servico e a mesma média. Porém há casos em que os clientes diferenciam-se quanto à distribuição de serviço, agrupando-se em classes, dentro das quais são homogêneos. Assim pode-se modelar uma fila como sendo constituida de uma ou mais classes de clientes. Ao conjunto dos elementos classes e estação de serviço dá-se também o nome de nodo.

4.2.1.3 O Modelo de rede de filas

O modelo de interesse neste trabalho considera múltiplas estações de serviço e suas classes, operando de maneira assíncrona e concorrente, o qual é chamado de rede de filas. Um exemplo de sistema real que pode ser assim modelado é um sistema de computação multiprogramado, onde o processador central (UCP) e os processadores de entrada e saída (PES) operam paralelamente.

Define-se como rede fechada de filas uma rede de filas consistindo de M nodos (estações de serviço e classes) interligados, numerados 1,2...M, com uma topologia de rede geral. Não há chegada externa de clientes e nem partida para o exterior. Assim a população é constante e finita (Ver a figura 4.2). Um cliente que deixa um nodo i, classe j vai para um nodo k, classe l com probabilidade p(i,j)(k,l).

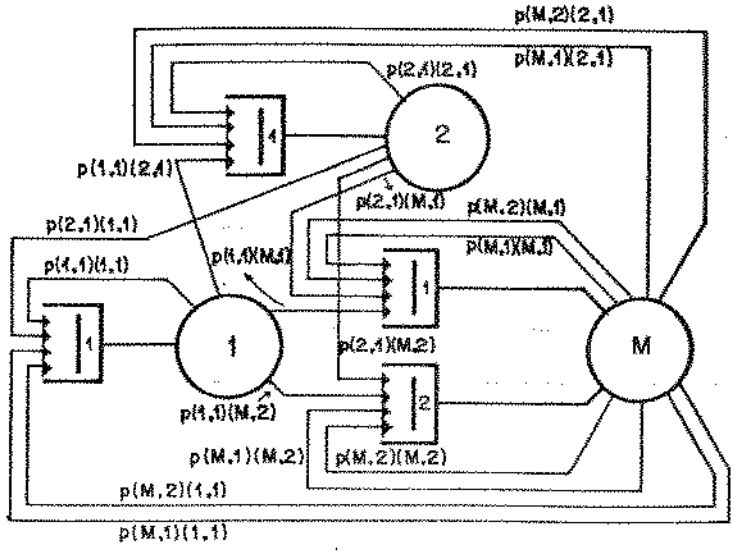


Figura 4.2 - Rede fechada de filas

Uma rede aberta de filas, além dos M nodos, contém uma "fonte" de população infinita e um "dreno" que absorve os clientes que partem da rede para o exterior. O fluxo de clientes oriundos da fonte é um processo de chegadas com taxa "lambda". Um cliente que chega da fonte vai para um nodo i,classe j com probabilidade p(f)(i,j). Um cliente que parte de um nodo i,classe j vai para um nodo k,classe l com probabilidade p(i,j)(k,l) e para o dreno com probabilidade p(i,j)(d).

4.2.2 Descrição de um Simulador

O simulador tem por objetivo geral simular o comportamento e coletar estatísticas de modelos de redes de filas, abertas ou fechadas, conforme definido na seção anterior. As características específicas dos modelos considerados pelo simulador são as seguintes (algumas delas ampliam a definição de redes de filas):

. possui M nodos;
. possui uma ou mais fontes, quando a rede for aberta, e nenhuma no caso contrário;
. possui um dreno quando a rede for aberta;
. a estação de serviço de um nodo pode ter um ou mais

servidores em paralelo;

. possui uma ou mais classes de clientes, sendo que as classes podem ter demandas de serviço distintas, com distribuição exponencial ou determinística (constante);

. a geração de clientes nas fontes obedece ao processo Poisson de chegadas;

. cada classe pode ter seus clientes divididos em categorias de prioridade. O escalonamento de clientes de uma classe segue a disciplina de prioridades para atendimento. Dentro de uma categoria o escalonamento é FCFS;

. a ocorrência de clientes nas categorias de prioridades é descrita percentualmente em cada classe;

. os destinos dos clientes de uma classe de um nodo são descritos por probabilidades. Clientes destinam-se a nodos ou ao dreno;

. os clientes gerados em uma fonte destinam-se especificamente à uma classe de um nodo.

As entradas para o simulador são apresentadas abaixo:

a) parâmetros de configuração da rede de filas (ou parâmetros do sistema)

- número de nodos
- número de servidores por nodo
- número de classes por nodo
- número de categorias de prioridade por classe
- número de fontes

b) variáveis de estado inicial da rede de filas

- número de clientes por nodo
- número de clientes por classe
- número de clientes por categoria de prioridade

c) variáveis de entrada

- intervalo médio entre chegadas de clientes por fonte
- tempo de serviço por classe e sua distribuição (u(i,j))
- destinos e probabilidades de destino de cada classe (pd(i,j)(k,l))

- destino dos clientes gerados em cada fonte
- percentuais de ocorrência de clientes por categoria de prioridade em cada classe

d) duração da simulação expressa em números de eventos. Dois tipos de eventos são considerados para efeito de contagem da duração. O primeiro é o término de serviço de um cliente. O segundo é a chegada a um nodo de um cliente originário de uma fonte.

As variáveis de saida, que medem o desempenho do sistema, descritas abaixo, são apresentadas por valores totais de cada nodo e por valores relativos de classe e prioridade:

U - utilizacão da estação de servico pelos clientes.
T - vazão ou quantidade de clientes servidos na unidade de tempo;
$\overline{L}$ - número médio de clientes em fila, inclusive os em serviço;
$\overline{Q}$ - tempo médio de espera em fila, inclusive tempo em serviço.
Lmax- máximo de clientes em fila;

O simulador foi desenvolvido segundo o enfoque de escalonamento de eventos (Fishman,1973). Um evento indica uma alteração do estado de determinada entidade. Especificamente, os eventos considerados são a chegada de um cliente a uma fila (nodo) e o término de serviço ou partida de um cliente de uma fila. O enfoque de escalonamento de eventos enfatiza a descrição detalhada dos passos que ocorrem quando um evento individual acontece.

4.2.2.1 Estruturas de dados

Os tipos principais de estruturas de dados usados no simulador são aqui apresentados. O primeiro é uma lista encadeada para conter informação sobre eventos a acontecer, representada na

figura 4.3. Cada elemento da lista contém a seguinte informação:

- momento de ocorrência do evento;
- tipo do evento, se partida ou chegada externa;
- localização do evento, em um nodo ou em uma fonte;
- classe do cliente, quando for partida;
- categoria de prioridade do cliente, quando for o caso;
- indicação do próximo elemento da lista.

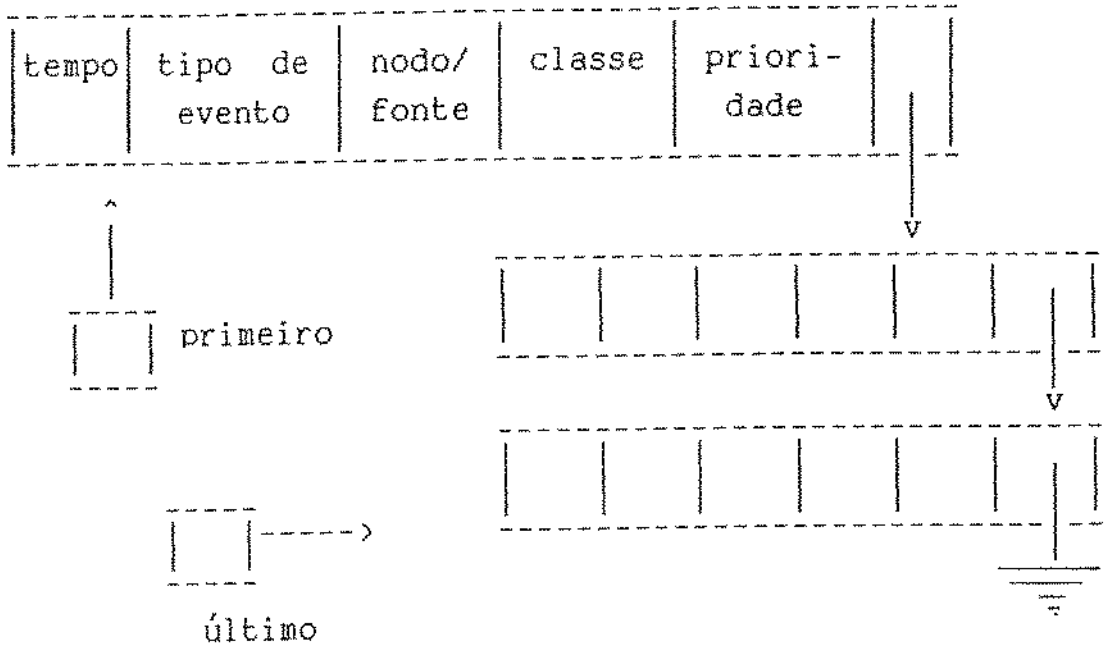


Figura 4.3 - Lista de eventos

Os elementos são encadeados segundo o campo tempo, em ordem crescente. Na operação de retirada, os campos do primeiro elemento são lidos e o elemento é devolvido a uma lista de espaço disponível. Na inserção, busca-se um elemento na lista de espaço disponível, seus campos são preenchidos e procura-se na lista o lugar adequado para inserção, conforme o campo de tempo. Convém observar que ao evento término de serviço em um nodo corresponde um evento de chegada em outro.

A segunda estrutura serve para registrar o estado corrente e o histórico das filas (total,classe e prioridade). Na figura 4.4 tem-se um esquema apresentando os diversos campos da estrutura. O estado corrente é representado pelo comprimento da fila e pelo

número de servidores ocupados. O histórico é registrado nos seguintes campos:

. comprimento máximo ocorrido;
. número de serviços completados ou clientes servidos;
. último instante de mudança de estado;
. somatório do produto tempo x comprimento;
. somatório do produto tempo x servidores ocupados.O registro de histórico tem como última finalidade permitir a elaboracão de estatisticas de desempenho.

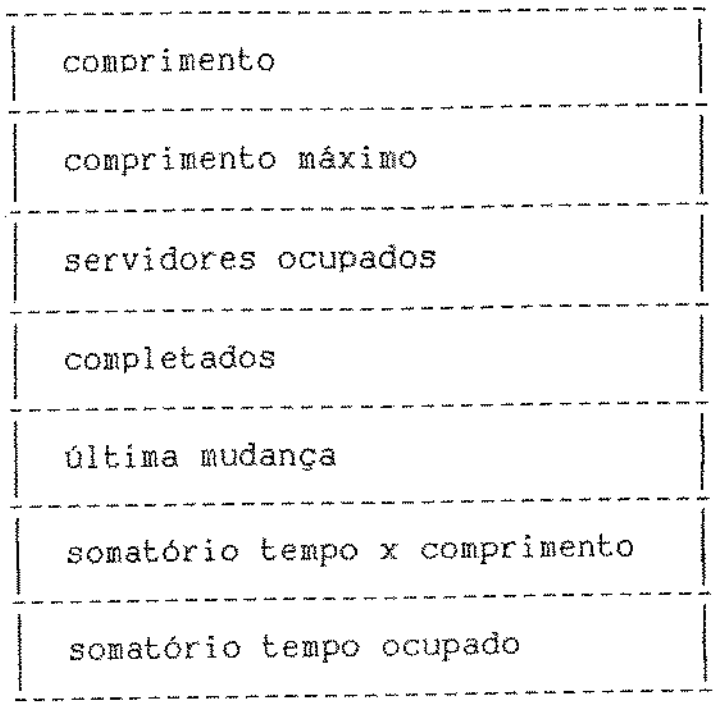


Figura 4.4 - Estrutura de dados de estado e histórico

A terceira estrutura de dados serve para representar os diversos nodos da rede, tanto nos aspectos estáticos como dinâmicos. Ver a figura 4.5. Usa-se um vetor de registros no qual cada um deles contém informação sobre um nodo. As classes de um nodo são descritas por outro vetor de registros. Como o modelo permite que o cliente de uma classe, após servido, tenha destinos variáveis, criou-se um vetor para especificação destes destinatários. Do mesmo modo, descrevem-se as categorias de prioridade. Deve-se observar que para cada entidade (nodo, classe, prioridade) está associada uma estrutura de representação do estado.

A última estrutura considerada serve para representar as fontes de clientes e é implementada de maneira semelhante às anteriores.

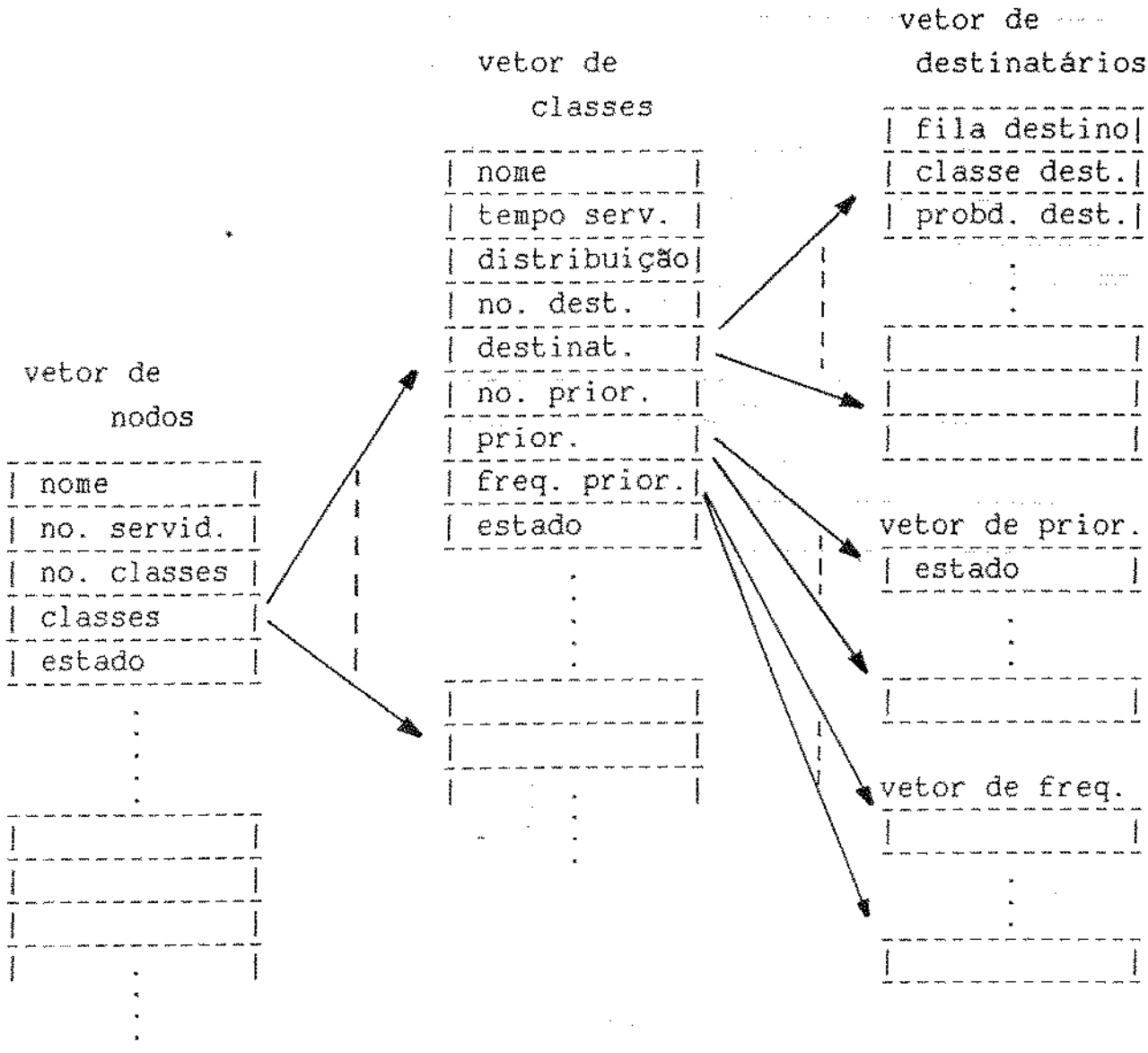


Figura 4.5 - Estrutura de representação da rede de filas

O conjunto formado pelo vetor de nodos e o vetor de fontes modela a rede de filas que se quer simular.

O projeto do simulador foi feito segundo a metodologia "Projeto Estruturado de Programas" (Myers, 1978), que se mostrou uma boa técnica auxiliar para confecção de programas.

4.2.2.2 Implementação e testes

A opção para implementação do simulador foi a favor do uso de uma linguagem de propósitos gerais, no caso o PASCAL. O ambiente inicialmente escolhido foi um microcomputador sob o sistema operacional CPM. Quanto à portabilidade do programa, procurou-se usar o mínimo possível de construções não constantes da definição original da linguagem. A atividade de simulação em computador demanda alto potencial de cálculo. Ao executar simulações típicas deste trabalho em microcomputador, o tempo de utilização tornou-se proibitivo. Assim partiu-se para implantação do programa em um computador de grande porte, no caso, um VAX 780. Isto foi feito sem grandes dificuldades. O tempo de simulação tornou-se bastante aceitável. Por exemplo, uma simulação de uma rede com 10 nodos, para 100 mil eventos, demanda aproximadamente 2 minutos. No microcomputador este tempo seria da ordem de 100 minutos.

Um aspecto importante a comentar é o da geração de números aleatórios. Normalmente existem duas opções: ou se faz uma rotina de geração ou se usa a rotina da biblioteca do compilador, quando houver. Escolheu-se para este simulador usar uma rotina própria do sistema. Foram feitos testes com a rotina e chegou-se a resultados satisfatórios de média e desvio padrão dos valores da sequência gerada, bem próximos dos valores teóricos.

Os algoritmos dos módulos do programa não serão aqui apresentados, por ser desnecessário.

O programa fonte, embora com um razoável grau de complexidade, ficou de tamanho pequeno, em torno de 700 linhas. Isto foi em parte devido às facilidades que a linguagem oferece, principalmente em termos de estruturas de dados. Por outro lado, procurou-se definir módulos no programa que fossem utilizados em várias partes do mesmo.

O programa foi testado quanto à correção dos resultados obtidos.

Como exemplo de teste, apresenta-se a seguir os resultados da simulação de uma rede de filas fechada para serem comparados com resultados teóricos e resultados obtidos por outro simulador da literatura (Sauer & Chandy,1981). O modelo consiste de duas filas: a primeira tem uma classe, um servidor, tempo médio de serviço igual a 6,666 com distribuição exponencial e destina os clientes para a outra fila. A segunda tem uma classe, dois servidores, tempo de serviço de 10,0 com distribuição exponecial e destina os clientes para a primeira fila. A população de clientes é 3. A simulação foi executada para 1000 eventos. Os resultados para comparação estão a seguir.

	fila	utilização	vazão	compr.médio	espera média
teóricos	1	0,812	0,122	1,596	13,082
	2	6,612	0,122	1,404	11,508
simulados	1	0,797	0,125	1,535	12,238
literatura	2	0,631	0,125	1,465	11,735
simulados	1	0,823	0,124	1,645	13,261
	2	0,589	0,124	1,355	10,971

4.3 O MODELO GERAL DE SDCD PARA ESTUDO

Nesta secão é definido um modelo de rede de filas de um SDCD para servir de base em estudos subsequentes. Assim como toda tarefa de modelagem, esta também adota uma série de simplificações e faz abstrações de detalhes do mundo real que se quer estudar. Os aspectos de interesse são os de comunicação, a qual é efetuada por troca de mensagens. Considera-se como ponto de partida o modelo de arquitetura da figura 3.2. A primeira simplificação manda que o nivel de aplicação consitirá apenas dos componentes ECL e SMO. Este último sendo composto por estações de monitoração e operação (EMO). Os componentes do modelo são:

no nivel APLIC tem-se (n) ECL's e (m) EMO's;

no nível SUCO tem-se (n+m) entidades, cada uma dando suporte a uma ECL ou EMO;

no nivel INTX tem-se um componente modelando os niveis inferiores de serviço de transporte de mensagens.

Cada componente de APLIC gera e fornece mensagens para SUCO e recebe e consome mensagens do mesmo. Cada componente de SUCO recebe mensagens de APLIC e de INTX e fornece mensagens para APLIC e INTX. Entidades de SUCO não geram nem consomem mensagens internamente neste modelo. O componente INTX recebe mensagens de SUCO e fornece mensagens para SUCO mas não gera nem consome mensagens internamente.

Toda geração de mensagens é independente das outras e segue o processo Poisson de chegadas. Todo serviço dado a uma mensagem em uma estação de serviço segue a distribuição exponencial.

As entidades de APLIC são modeladas como uma fonte geradora de mensagens, um nodo composto de uma estação de serviço mais fila e um dreno para consumo de mensagens. Ver a figura 4.6. Uma fonte de uma ECL gera dois tipos de mensagens:

- com informação de estado de variáveis e de sincronização, aqui tratadas conjuntamente como mensagens de controle. Estas se destinam à própria ECL geradora ou a ECL's remotas;

- com informação de monitoração, destinadas às entidades EMO.

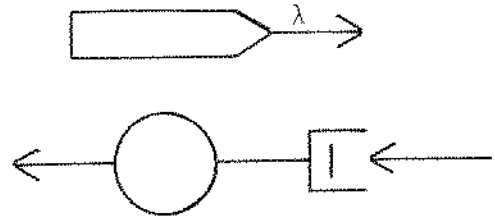


Figura 4.6 Modelo de entidade de APLIC

Definindo

$\lambda c \triangleq$ taxa de geração de mensagens de controle.

$\lambda m \triangleq$ taxa de geração de mensagens de monitoração,

e considerando que os dois tipos de mensagens são gerados independentemente, pode-se dizer que a taxa de geração de mensagens em uma ECL é

$$\lambda c + \lambda m$$

Definindo

$\alpha \triangleq$ proporção de mensagens de controle destinadas a ECL's remotas,

tem-se que a taxa de mensagens geradas em uma ECL e que têm destino remoto é dada por

$$\alpha\ \lambda c + \lambda m$$

A taxa de mensagens de controle geradas localmente e que retornam para a ECL geradora é

$$(1-\alpha)\ \lambda c$$

Uma fonte EMO gera mensagens de operação para serem enviadas às ECL's. Define-se

$\lambda o \triangleq$ taxa de geração de mensagens de operação.

A taxa de mensagens de operação que chegam a uma ECL é

$$(m/n)\ \lambda o$$

A taxa total de mensagens que chegam a uma ECL é

$$\lambda c + (m/n)\ \lambda o$$

A taxa total de mensagens que chegam a uma EMO é

$$(n/m)\ \lambda m$$

As entidades do SUCO são modeladas como um nodo composto de uma estação de serviço com espaço para duas classes de mensagens (duas filas). Uma classe consiste das mensagens que chegam de APLIC e outra consiste das mensagens que chegam de INTX. Mensagens que chegam de APLIC, após o serviço são destinadas a INTX com probabilidade (p); são destinadas para APLIC com com probabilidade (1-p). Mensagens oriundas de INTX destinam-se sempre a APLIC (Ver a figura 4.7). Aqui ocorrem simplificações em relação ao SUCO real: não foi considerada a existência de mensagens que se destinam ao próprio SUCO. Isto porque, em regime, a proporção deste tipo de mensagens pode ser considerada desprezivel.

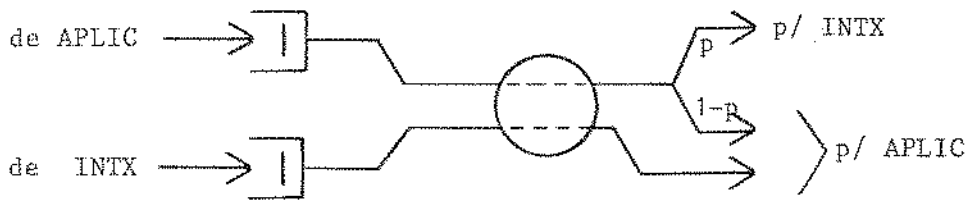


Figura 4.7 Modelo de entidade de SUCO

O INTX completo é modelado como um nodo composto de uma estação de serviço e (n+m) filas para acomodar as mensagens oriundas do SUCO. Ver a figura 4.8.

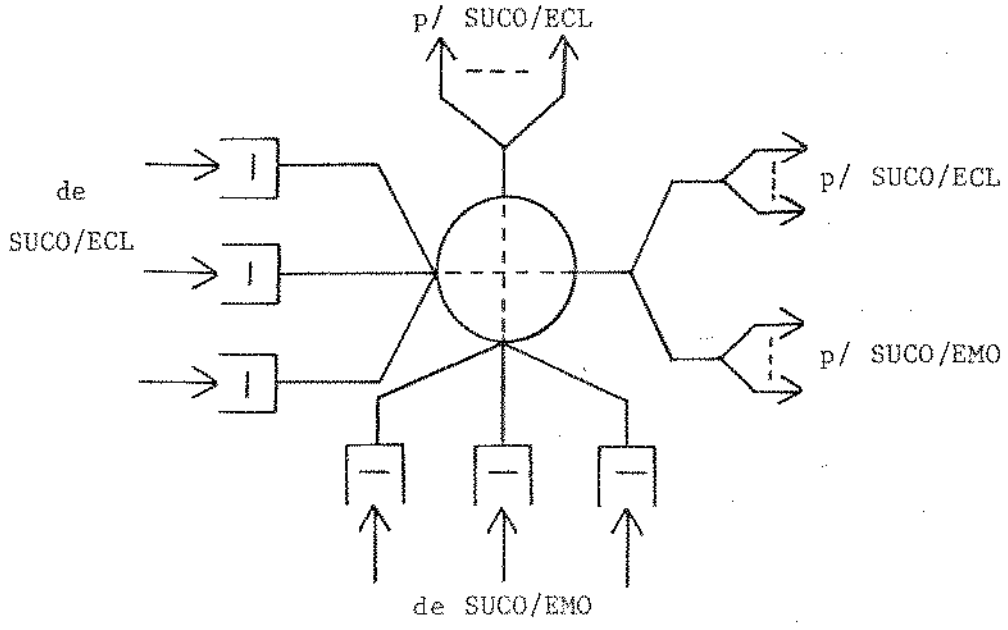


Figura 4.8 Modelo de INTX

Mensagens que chegam de um nodo SUCO provenientes de uma ECL têm os seguintes destinos:

se for mensagem de controle, destina-se a qualquer nodo SUCO/ECL com igual probabilidade;

se for mensagem de monitoração, destina-se a qualquer nodo SUCO/EMO com igual probabilidade.

Mensagens que chegam de um nodo SUCO provenientes de uma EMO destinam-se a qualquer nodo SUCO/ECL com igual probabilidade.

O modelo assim criado servirá como ponto de partida para subsequentes estudos por simulação que serão feitos.

4.4 ESTUDO DE DESEMPENHO FIM_A_FIM

Tendo como ponto de partida o modelo geral de estudo apresentado na seção anterior, busca-se aqui o levantamento de medidas de desempenho de um exemplo de SDCD, através de um modelo simples. Em particular, as medidas de interesse são:

atraso médio imposto a uma mensagem, quando de sua transmissão de uma entidade da aplicação a outra situada remotamente;

atraso médio imposto a uma mensagem, quando de sua transmissão de uma entidade da aplicação a outra situada localmente.

O modelo em questão considera os seguintes componentes:

nodos de APLIC/ECL, com respectivas fontes de mensagens;
nodos de SUCO;
nodo de INTX.

O modelo de rede aberta de filas está mostrado esquematicamente na figura 4.9. Cada componente APLIC é modelado, por um lado, como uma fonte de mensagens para o SUCO e, por outro lado, como uma fila e uma estação de serviço que consome mensagens. Cada componente SUCO é modelado como uma estação de serviço e duas filas ou classes. O INTX é modelado como uma estação de serviço e três filas ou classes.

Com relação às caracteristicas dos componentes do modelo geral de estudo, as seguintes considerações e simplificações foram feitas:

o nivel de APLIC contém apenas ECL's;

as mensagens são do tipo "controle", como consequência imediata da colocação anterior. Estas destinam-se pois à própria ECL geradora ou às ECL's remotas;

a proporção de mensagens gerada em cada ECL e que se destinam às ECL remotas é dada pelo fator α. Consequentemente, a proporção de mensagens que retornam à ECL geradora é dada por $(1-\alpha)$;

mensagens que chegam à fila de uma ECL têm prioridades assinaladas. Foram consideradas duas categorias de prioridades. As proporções de mensagens em cada categoria de prioridade são dadas por p e 1-p;

o nivel de SUCO fica como no modelo geral, com uma entidade associada a cada ECL. Existe uma classe de mensagens oriundas de APLIC e outra classe de mensagens oriundas de INTX;

o nivel de INTX tem uma estação de serviço e uma fila para cada entidade do SUCO. Trata mensagens segundo duas categorias de prioridade, as quais ocorrem segundo as proporções p e 1-p. O destino das mensagens oriundas de uma entidade SUCO e servidas por INTX se distribui equiprovalvelmente entre as outras entidades restantes do SUCO.

O objetivo primeiro deste estudo por simulação é verificar o atraso fim-a-fim de mensagens, por categoria de prioridade, em função da taxa de geração de mensagens nos nodos APLIC.

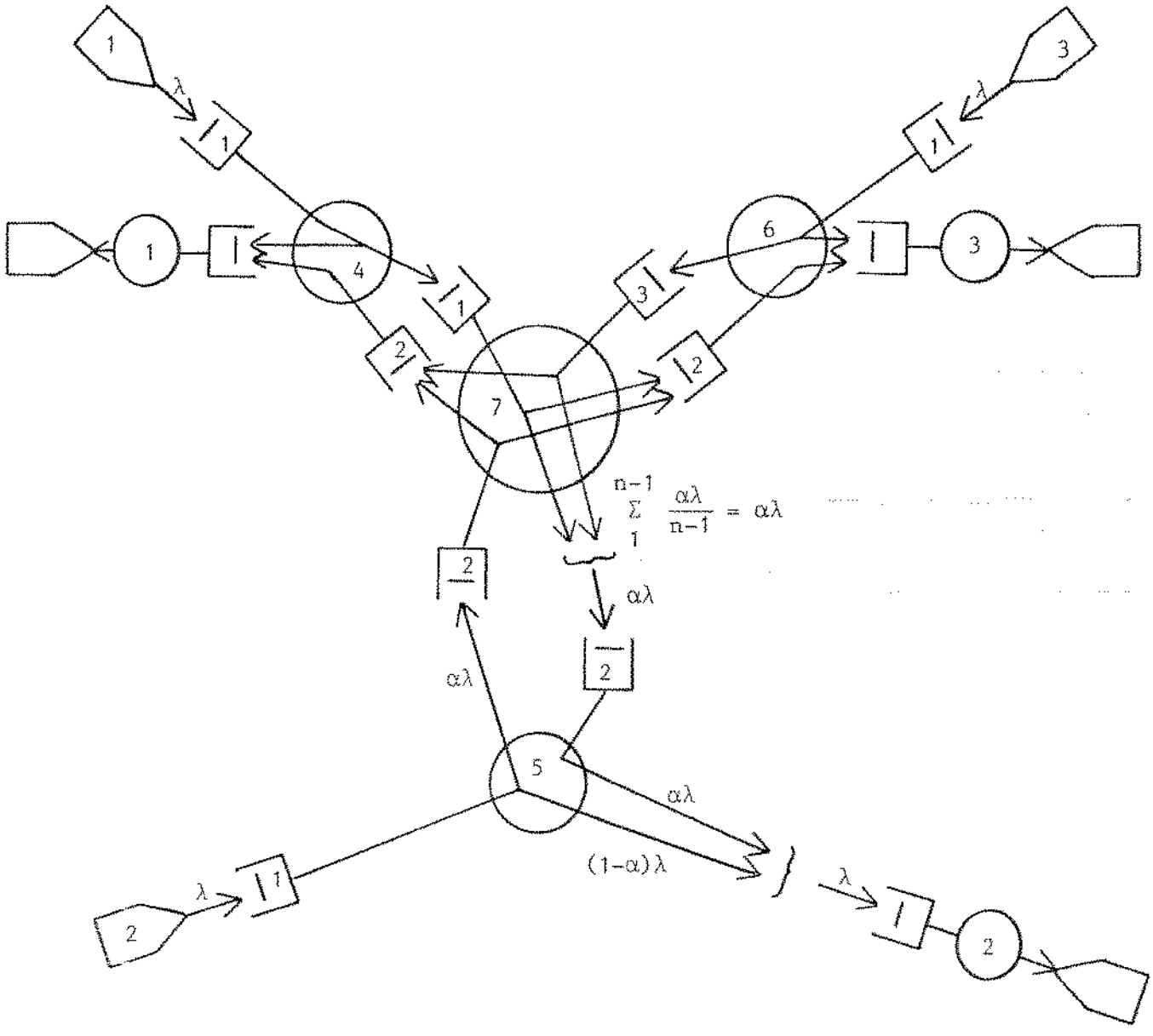


Figura 4.9 Modelo de rede aberta

4.4.1 Caracterização dos Componentes

A seguir são estabelecidos os parâmetros e definidos os valores para as variáveis do modelo de rede de filas para a simulação, a partir de considerações e hipóteses acerca de aplicações e sistemas de comunicação do mundo real. O ponto de partida para o levantamento dos valoress se constitui do assunto tratado na seção 3.4.1, quando da elaboração de propriedades quantitativas requeridas para o SUCO. Da mesma forma, consideram-se as caracteristicas de comunicação, a nivel de aplicação, tratadas na seção 3.2 e em outras partes deste trabalho.

O componente APLIC é concebido como um conjunto de processos controladores que executam ciclicamente suas funções. Em cada ciclo os controladores enviam e recebem mensagens, efetuam o controle e adquirem novos dados dos subprocessos fisicos sendo controlados.

Seja, para o caso de geração de mensagens,

C = número de controladores
i = 1,2,....,C o i-ésimo controlador
t(i) = valor em unidades de tempo do ciclo do controlador i
m(i) = número médio de mensagens enviadas/ciclo/controlador

Considerando que a geração de mensagens em cada controlador é independente das outras e segue o processo de Poisson de chegadas, tem-se para o controlador i:

$$\lambda(i) = \frac{m(i)}{t(i)} \qquad e \qquad \lambda = \sum_{1}^{C} \lambda(i),$$

que é a taxa de geração de mensagens em cada entidade APLIC.

Assume-se para efeito deste estudo, que

C = 1Ø,
i = 1,2,...1Ø,

t(i) = Ø,2 segundos,
e m(i) é variável.

A taxa de mensagens gerada é dada então por:

$$\lambda(i) = \frac{m(i)}{Ø,2}$$

$$\lambda = \sum_{I}^{1Ø} 5m(i) = 5Øm(i)$$

Esta taxa será a variável da simulação, função de m(i). A proporção de mensagens com destino remoto foi escolhida arbitrariamente como sendo

$$\alpha = 1 - \alpha = Ø,5Ø$$

Para o caso de consumo de mensagens por APLIC, modela-se, por questões de simplicidade, todos os controladores em conjunto como um nodo composto de uma estação de serviço, com um servidor exponencial e uma fila atendida segundo as prioridades das mensagens.

Seja,

c = capacidade média de consumo de mensagens por ciclo, e

t = valor do ciclo.

Assumindo que,

c = 3Ø msg/ciclo, e

t = Ø,2 segundos,

tem-se a taxa de serviço

$\mu = c/t$ ou $\mu = 15Ø$ msg/s.

Existem duas categorias de prioridade, com as seguintes proporções de mensagens:

$$p = 1 - p = 0,5.$$

Na modelagem das entidades do SUCO é necessário apenas estimar as taxas de serviço, que foram escolhidas iguais para todas as entidades e também iguais para as duas classes de cada entidade. Esta estimativa é feita, para este caso, baseando-se nos valores médios das taxas de chegada de clientes no nodo e em um coeficiente de utilização desejado para o nodo. Cumpre observar que estas taxas de serviço poderiam ser obtidas de outras formas. Por exemplo, através da capacidade de processar mensagens, de um determinado processador no qual estivessem implementados os serviços do nível de SUCO.

As mensagens que chegam ao SUCO são, na média, dadas pela seguinte expressão:

$$\lambda s = \lambda + \alpha \lambda.$$

O primeiro termo à direita do sinal representa as mensagens oriundas de APLIC. O segundo termo representa as mensagens oriundas de INTX, onde α é a proporção de mensagens que têm destino remoto.

Assim, para λ valendo tipicamente 100msg/s, α = 0,5 e um coeficiente de utilização de 75%, tem-se:

$\rho = \lambda s / \mu$, o que dá μ = 200msg/s.

Na modelagem de INTX é necessário apenas estimar a taxa de serviço, que deve ser coerente com os valores usuais em redes locais. Da mesma forma,

$$\lambda i = \sum_{1}^{3} \alpha \lambda = 1,5 \lambda.$$

Escolhendo um coeficiente de utilização de 75% e com o mesmo λ anterior, tem-se

$\rho = \lambda i / \mu = 0,75$, o que dá μ = 200msg/s.

4.4.2 Simulação do Modelo

O modelo acima estabelecido foi submetido ao simulador, tendo como variável a taxa de geração de mensagens em APLIC. As medidas de desempenho obtidas, em particular os tempos médios de espera em fila e no sistema, são fornecidos para cada nodo. Assim, para determinar o tempo médio em uma determinada rota é necessário calcular o somatório dos tempos nos nodos pertinentes, tomando-se os valores para cada classe e categoria de prioridade nas quais uma mensagen se insere no decorrer de sua trajetória.

O atraso para uma mensagem que é transportada de um local para outro da rede é calculado como se segue. Seja a tripla (i,j,k) a referência do nodo i, classe j, prioridade k.

Seja

T (i,j,k) $\triangleq$ tempo no sistema de filas, e

W (i,j,k) $\triangleq$ tempo em fila .

Quando não existe classe ou categoria de prioridade, o índice correspondente vale Ø. Por exemplo, o tempo médio (ou atraso) decorrido na transferência de mensagens de prioridade 1, geradas na ECL de um determinado elemento de computação, representado pela fonte número 1, para outra ECL remotamente situada, representada pelo nodo número 3, é dado por:

$$A = T(4,1,Ø) + T(7,1,1) + T(6,2,Ø) + W(3,1,1).$$

Uma mensagem chega ao seu destino quando recebe serviço do nodo destinatário. Esta é a razão de se considerar apenas o tempo médio em fila no nodo APLIC, de número 3.

O tempo médio decorrido entre transferências locais é dado, no

caso da fonte número 1 e do nodo número 1, por:

$$A = T(4,1,0) + W(1,1,1).$$

Os resultados assim obtidos estão mostrados no gráfico da figura 4.10. Na mesma figura há também os resultados obtidos analiticamente que, para o modelo escolhido, foi possível determinar. A formulação analitica será mostrada adiante.

Pode-se ver pela figura que os pontos levantados pela simulação determinam curvas que se aproximam bastante das curvas obtidas por formulações analiticas. O atraso imposto às mensagens cresce com o crescimento da taxa de mensagens em cada ECL. Para a transmissão fim-a-fim, o atraso cresce muito rapidamente a partir de 80 msg/s, indicando uma carga que o sistema não pode suportar com eficiência. O mesmo ocorre com o atraso local, a partir de 100 msg/s. Os valores razoáveis, para o modelo definido, são de até 40 msg/s por ECL para o caso fim-a-fim, que dão atrasos de até 25 ms. Para o caso local, com 60 msg/s por ECL, obtém-se atrasos em torno de 10 ms. Deve ser notado que mesmo estas taxas de mensagens são altas para a maioria das aplicações de controle em tempo real. Considerando, por exemplo, a taxa de 50 msg/s por ECL, esta deveria gerar uma mensagem a cada 20 ms em média. Uma dessa ECL controlando 10 pontos de maneira cíclica e, a cada ciclo, enviando uma mensagem, deveria ter para o ciclo o período de 200 ms. Este é um valor baixo para a maioria das aplicações, principalmente do ponto de vista da monitoração e da operação remotas.

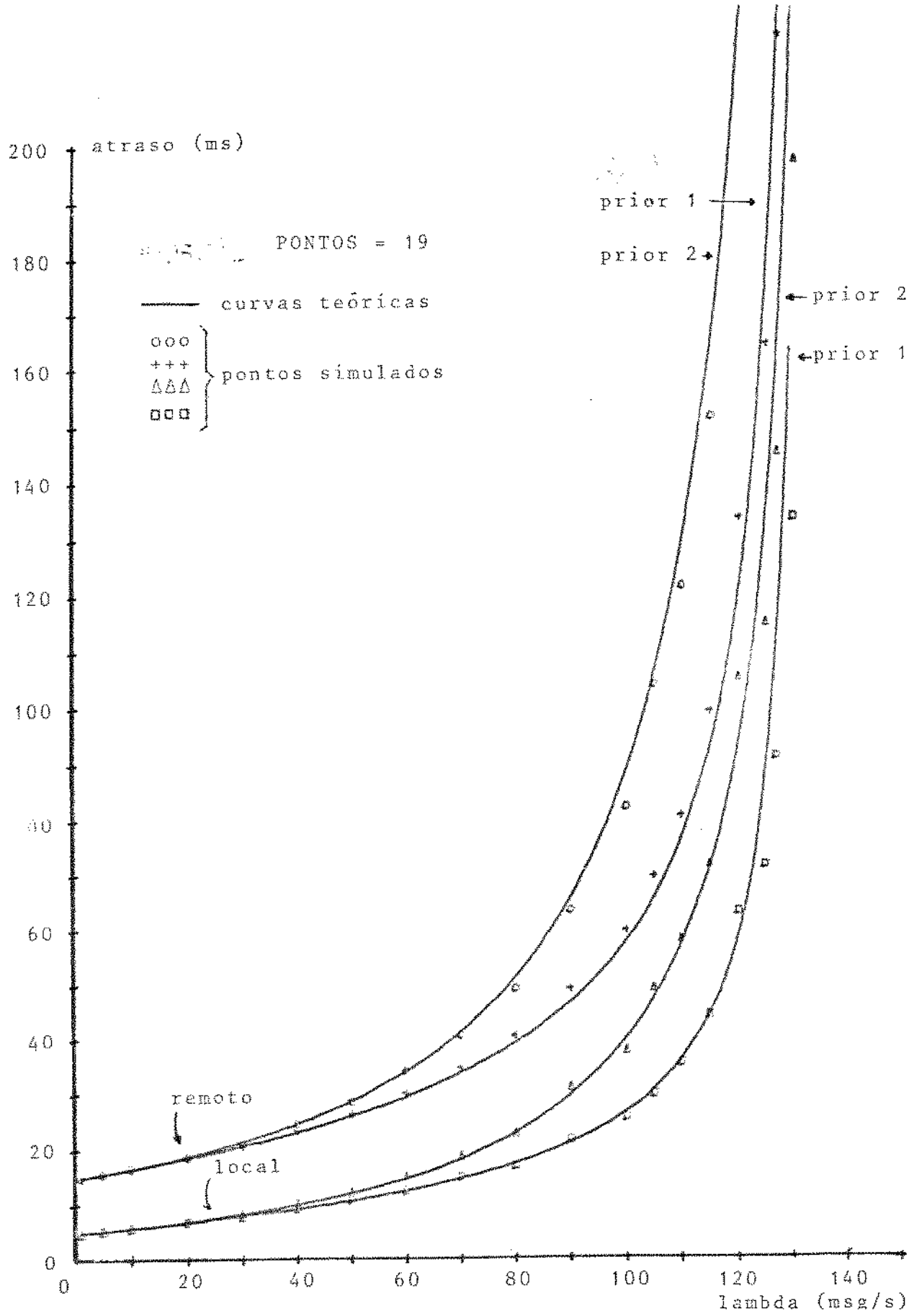


Figura 4.10 Atraso fim-a-fim

4.4.3 Verificação Analitica do Desempenho

Busca-se, nesta seção, obter analiticamente os resultados de desempenho do modelo anteriormente definido, para comparação com os resultados obtidos na simulação. As medidas procuradas são os atrasos na transmissão de mensagens. O método usado para se chegar aos resultados finais é o estudo isolado de cada sistema de filas que compõe a rede. As variáveis em cada sistema de filas são as taxas de chegada. Para determiná-las é necessário encontrar seus valores médios e suas distribuições. A partir dos resultados parciais de atraso, o resultado total para cada trajetória é encontrado, como no caso da simulação.

Baseando-se nos estudos de Jackson em 1957 (Apud Kleinrock, 1975a), e em outros autores posteriores, chega-se ao modelo de cada nodo isoladamente. O sistema estudado por Jackson consiste de uma rede de filas com N nodos, onde o i-ésimo nodo contém mi servidores exponenciais com taxa μi; ademais, o i-ésimo nodo recebe chegadas do exterior do sistema na forma de um processo de Poisson com taxa γi. Após deixar o i-ésimo nodo, um cliente vai para o j-ésimo nodo com probabilidade rij, onde rij >= Ø. Após completar serviço no nodo i, a probabilidade de um cliente deixar a rede é dada por

$$1 - \sum_{j=1}^{N} rij.$$

Deve ser calculado, para cada nodo, sua taxa média de chegada. Para tal, somam-se as chegadas (Poisson) oriundas do exterior mais as chegadas (não necessariamente Poisson) oriundas de todos nodos internos. Ou seja, denotando a taxa média de chegada no nodo i por λi, o conjunto de parâmetros deve satisfazer às seguintes equações:

$$\lambda i = \gamma i + \sum_{j=1}^{N} \lambda j \; rij \qquad i=1,2...N$$

Um ponto de partida básico para se chegar a esta conclusão consiste dos resultados apresentados por Burke (Apud Kleinrock, 1975a): "a saida de uma fila M/M/m, no estado estacionário, com parâmetro de entrada λ e parâmetro tempo de serviço μ, para cada um dos m servidores é um processo Poisson com a mesma taxa λ".

Estudos posteriores, referenciados por Sauer & Chandy (1981), estenderam estes resultados para redes com outros tipos de nodos. De particular interesse, aqui, são os nodos com classes de clientes com escalonamento FCFS, mesmos tempos médios de serviço e distribuição exponencial.

O nodo SUCO contém 2 classes, com escalonamento FCFS, o mesmo tempo médio de serviço e distribuição exponencial. Conforme as colocações anteriores, se a entrada para ele for Poisson, a saída também o será.

O nodo INTX tem 3 classes; em cada uma delas os clientes são escalonados para serviço conforme prioridades atribuídas externamente no momento da chegada. Considerando que (1) em cada categoria de prioridade de uma classe os clientes chegam segundo o processo de Poisson; (2) o tempo de serviço tenha distribuição exponencial de mesma média, pode-se dizer que a saída global de uma classe é também Poisson. Ademais, se os clientes das 3 classes são servidos com o mesmo tempo médio, pode-se afirmar que a saida global do nodo também é Poisson.

Para a fila APLIC o raciocínio é o mesmo. Assim, conforme Jackson pode-se proceder à análise separada de cada nodo do modelo. Passa-se então à formulação analitica dos resultados para cada tipo de nodo.

Nodo SUCO

Seja um nodo a um servidor que contenha as seguintes características:

C = número de classes;
λc = taxa de chegada da classe, Poisson;

$\lambda = \sum_{c} \lambda c_i$

μ = taxa de serviço para todas as c classes, exponencial.

Sejam as seguintes medidas de desempenho:

ρc = utilização do servidor pelos clientes da classe c;
ρ = utilização do servidor por todos os clientes;
Nc = no. médio no nodo de clientes da classe c;
Tc = tempo médio dispendido no nodo p/ clientes classe c;
Wc = tempo médio de espera em fila p/ clientes classe c;

Segundo Sauer & Chandy (1981), as formulações para as medidas são dadas pelas equações:

$$\rho c = \lambda c / \mu \qquad e \qquad \rho = \sum_{c} \rho c$$

$$Nc = \frac{\rho c \ \rho}{1 - \rho} + \rho c$$

$$Tc = \frac{\rho}{\mu} \ \frac{1}{1 - \rho} + \frac{1}{\mu}$$

$$Wc = \frac{\rho}{\mu} \ \frac{1}{1 - \rho}$$

Para o nodo SUCO, com C=2 e as taxas de chegada mostradas na figura 4.11 tem-se.

$$\rho 1 = \lambda a / \mu \qquad \rho 2 = \alpha \lambda a / \mu \qquad \rho = (1+\alpha) \lambda a / \mu$$

$$N1 = (1+\alpha) \frac{\rho 1^2}{1 - \rho} + \rho 1 \qquad N2 = \alpha (1+\alpha) \frac{\rho 1^2}{1 - \rho} + \rho 1$$

$$T1 = T2 = \frac{\rho}{\mu} \ \frac{1}{1 - \rho} + \frac{1}{\mu}$$

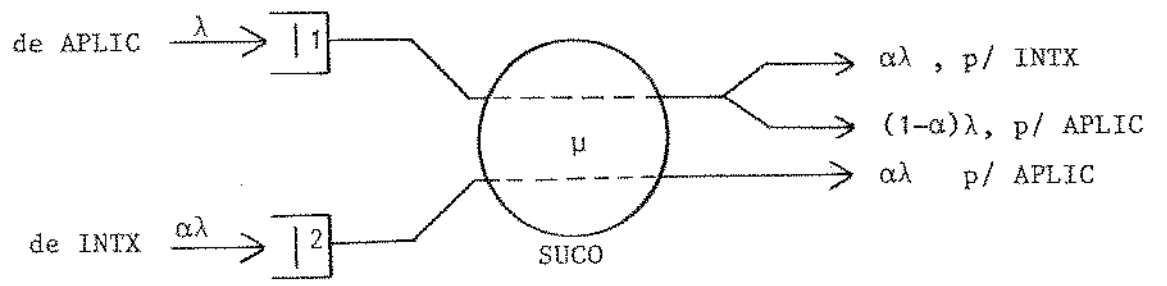


Figura 4.11 Nodo SUCO

Nodo APLIC

No estudo individual do nodo APLIC, busca-se as medidas de desempenho relativas a cada categoria de prioridades. Este tipo de sistema de filas denomina-se HOL, de "head-of-the-line", e foi estudado por Cobham em 1954, como mostra Kleinrock (1975b). A figura 4.12 esquematiza este tipo de sistema de filas. Os clientes são separados com base no grupo de prioridade a que pertençam. Sendo p=1,2,....P a identificação de um grupo ou categoria de prioridade, o sistema escalona clientes do grupo que no instante do escalonamento tenha maior p. Dentro de um mesmo grupo a política é FCFS.

Considerando o caso em que: ocorrendo a chegada de um cliente em um grupo, seja p2, está sendo servido um cliente de um grupo, seja p1, tal que p2 > p1; o cliente em serviço não é retirado do servidor (escalonamento não preemptivo). O tempo médio em fila para os membros do p-ésimo grupo é dado por:

$$Wp = \frac{W\emptyset}{(1-\sigma p)(1-\sigma p+1)} \qquad p=1,2....P.$$

onde $\sigma p = \sum_{i=p}^{P} \rho i$

$\rho i = \lambda i \ \overline{xi}$

$W\emptyset = \sum_{i=1}^{P} (\lambda i \ \overline{x^2 i}) \ / \ 2$

$\overline{xi^2}$ = segundo momento do tempo de serviço para clientes do grupo i.

No caso do sistema de fila APLIC, com

$P = 2,$

$\lambda = \lambda 1 + \lambda 2$, Poisson,

$\lambda 1 = p\lambda$ $\lambda 2 = (1-p)\lambda,$

$\overline{xi} = \overline{x}$, distribuição exponencial,

tem-se

$$W\emptyset = (\lambda 1 + \lambda 2) \ / \ \mu^2 = \lambda \ / \ \mu^2$$

$$\sigma 1 = \rho 1 + \rho 2 = (\lambda 1 + \lambda 2) \ / \ \mu = \lambda \ / \ \mu$$

$$\sigma 2 = \rho 2 = \lambda 2 \ / \ \mu$$

$$W1 = (\lambda \ / \ \mu^2) \ \frac{1}{(1-\rho)\ (1-\rho 2)}$$

$$W2 = (\lambda \ / \ \mu^2) \ \frac{1}{(1-\rho 2)}$$

O tempo médio no sistema é dado por

$$Ti = Wi + 1 \ / \ \mu.$$

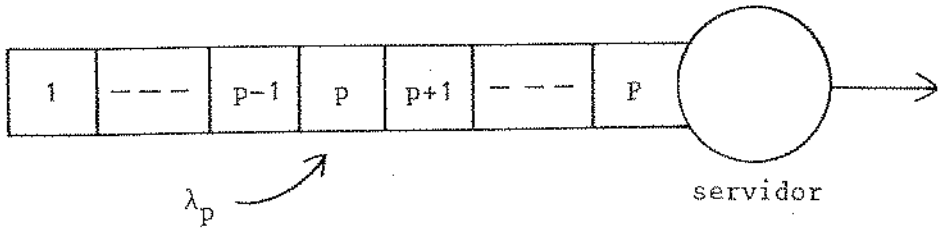


Figura 4.12 Nodo tipo HOL

Nodo INTX

O nodo INTX, embora tenha 3 classes com prioridades, pode ser simplificado para uma só classe. A taxa de chegada em cada grupo p de cada classe tem o mesmo valor médio com distribuição exponencial. O atendimento dos clientes do p-ésimo grupo, considerando todas as classes é FCFS. A taxa de serviço é a mesma para todos os clientes e tem distribuição exponencial. Assim pode-se considerar que o nodo tem uma só classe do tipo HOL; a taxa de chegada para o p-ésimo grupo da classe resultante é a soma das taxas de chegada dos p-ésimos grupos correspondentes, em cada classe individual. Isto está mostrado na figura 4.13.

As equações que dão os tempos médios de espera em fila e no sistema são semelhantes às do nodo APLIC. Para encontrar o atraso médio fim-a-fim teórico, como mostrado no gráfico da figura 4.10, foram utilizadas as mesmas equações da seção 4.4.2.

O levantamento de medidas de número médio de clientes em cada nodo segue raciocínios semelhantes, que aqui não são mostrados afim de não se fugir demais do objetivo geral do trabalho e também devido à exiguidade de tempo e espaço.

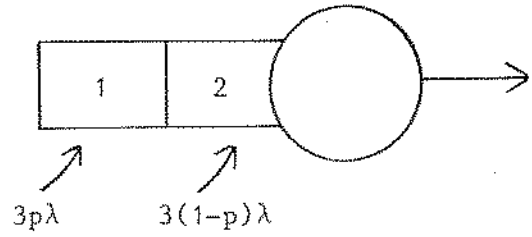


Figura 4.13 Nodo INTX simplificado

4.5 CONCLUSOES

Neste capitulo foi elaborado um modelo de rede de filas para sistemas distribuidos e apresentado um programa simulador. Foi executada a simulação de um exemplo especifico para se obter dados de desempenho. Os resultados foram verificados através de cálculos com formulações analiticas da teoria de filas. Os valores coincidiram, o que é um bom indicativo para validar o simulador. Além disso indicam que, para modelos da mesma classe do modelo simulado, os dados de desempenho podem ser obtidos com um custo menor através de formulações matemáticas. A simulação por computador em geral é cara. Porém, para modelos de maior grau de complexidade, maior número de componentes e, principalmente, com variáveis de distribuição diferente da exponencial, a análise matemática se torna altamente complexa. Nesses casos, a simulação por computador se torna mais vantajosa.

O objetivo primordial do capitulo foi elaborar ferramentas de análise de desempenho para sistemas distribuídos, no aspecto de

transferência de mensagens, em particular para sistemas orientados a aplicações de controle distribuído de processos. As ferramentas são o modelo de redes de filas e o simulador, com as quais pretende-se facilitar as atividades de projeto, especificação e avaliação de sistemas, como exposto na introdução ao capítulo.

Não é objetivo deste trabalho o tratamento extensivo deste tema. Por isso não foram feitas outras análises. Um estudo importante e interessante de ser feito é o do comportamento interno ao SUCO, ou seja, o estudo do fluxo de mensagens entre as estruturas de dados internas (filas de dados). Nesta análise tem-se que considerar uma disciplina de escalonamento do tipo compartilhamento de processador (PS - "processor sharing"). Isto implica em algumas modificações no simulador, o que sem dúvida aumentaria seu potencial e generalidade.

5 CONCLUSÕES GERAIS

Neste trabalho buscou-se contribuir em alguns aspectos de sistemas distribuídos para automação industrial, principalmente no tocante à comunicação. O trabalho se deu no sentido de caracterizar os requisitos de comunicação das aplicações e, em vista disso, elaborar proposições concretas para dar subsídios à concepção de mecanismos de suporte a esta comunicação. Para isso foram levantados os requisitos de comunicação de aplicações particulares (SDCD); foi elaborado um modelo de programas paralelos da aplicação e especificado, até o nível de implementação, um subsistema de suporte à comunicação e sincronização entre processos computacionais comunicantes. Além disso, tratou-se da análise de desempenho de sistemas distribuídos, para a qual foi elaborado um programa simulador de rede de filas.

Durante o período de desenvolvimento deste trabalho, foram geradas algumas publicações e apresentações de trabalhos em congressos. Estas publicações se mostraram importantes para o próprio desenvolvimento do trabalho porque serviram de base para discussões e troca de idéias, com consequente reflexos positivos para o mesmo. Foram publicados dois relatórios técnicos (Nogueira, 1984-a e 1984-b), três trabalhos em congressos (Nogueira & Mendes & Ruggiero, 1982; Nogueira & Lages, 1983; Nogueira & Ruggiero & Mendes, 1983). Um artigo foi recentemente publicado na revista Controle & Instrumentação (Castilho & Nogueira, 1985).

Um ponto que deve ser mencionado é sobre a possibilidade de implementação do SUCO num ambiente que tenha um subsistema de interconexão real e padronizado, como o da proposta do Comitê 802 do IEEE, para redes locais. Nesse padrão, o protocolo do nível de enlace lógico é semelhante ao HDLC. Já foi dito anteriormente que para poder definir o SUCO foi necessário especificar uma camada inferior, incluindo os serviços e a interface. Os serviços do

subsistema de interconexão mais indicados para aplicações de tempo real são baseados em datagramas, como considerado no capítulo 3. A interface, todavia, é variável, e depende de implementações particulares. A flexibilidade necessária para que isso possa ser feito deve ser provida em qualquer especificação. No caso do SUCO, as modificações dar-se-ão principalmente nos parâmetros das primitivas envia e recebe, uma tarefa que não tem muitas implicações.

Todavia pode ocorrer , em determinadas implementações, que o chamador da primitiva esteja no nível inferior, diferentemente da proposta inicial do SUCO. Na especificação atual do SUCO, um procedimento de execução cíclica pede mensagens do nível inferior através da primitiva recebe e as insere na fila de entrada. No outro caso, este procedimento deverá ser chamado pelo nível inferior a cada transferência. Da mesma forma acontece com o procedimento que retira mensagens da fila de saída e as entrega ao nível inferior através da primitiva envia.

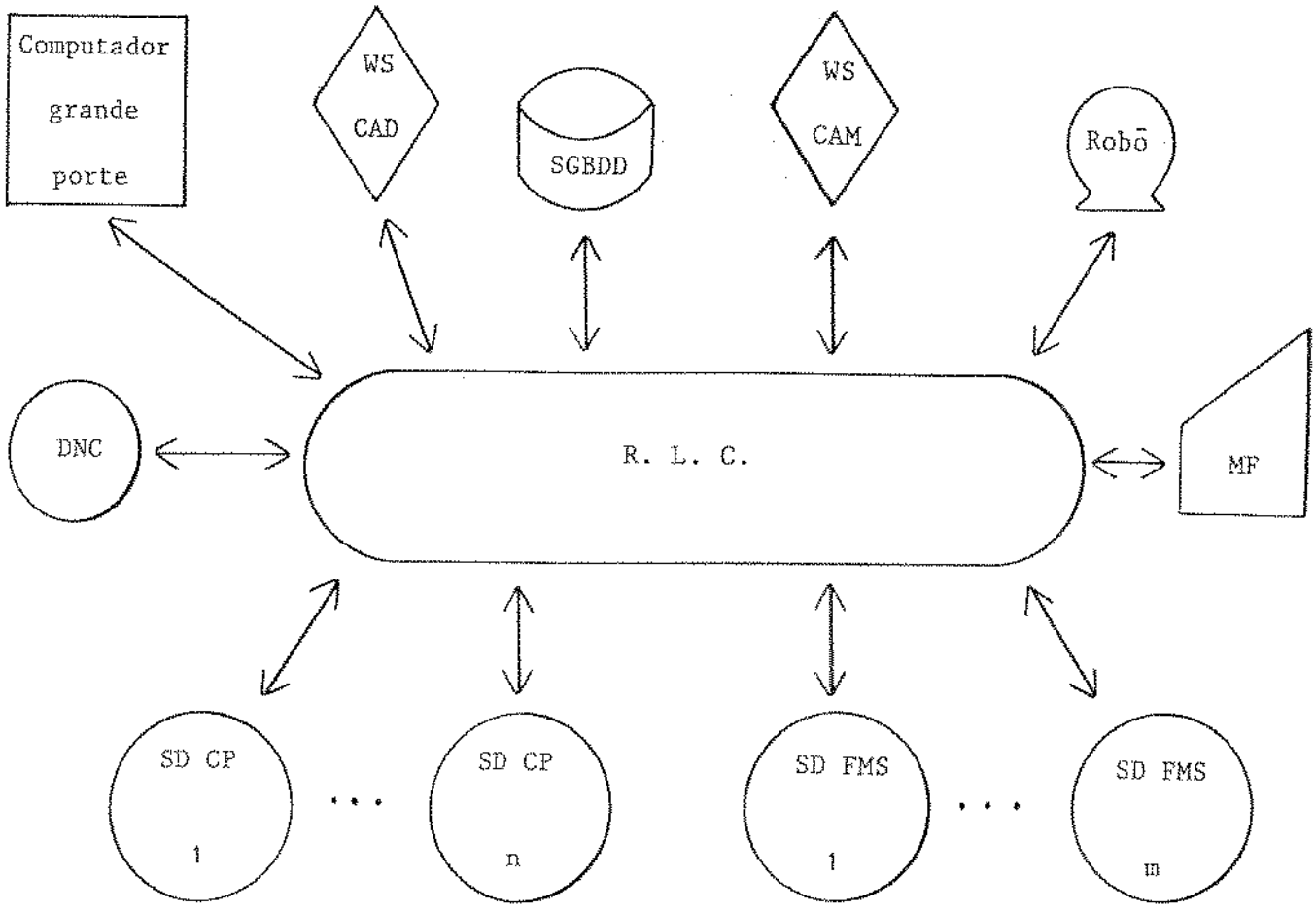
No capítulo 4 foi apresentada uma metodologia para auxílio no projeto, especificação e avaliação de sistemas distribuídos, tomando-se como exemplo aplicações de SDCD's no âmbito de comunicação fim-a-fim. Por falta de tempo e espaço não foi possível apresentar outros exemplos de análise de desempenho. Assim, também neste tópico, há a possibilidade de continuação do trabalho.

A caracterização das aplicações de automação industrial em ambiente distribuído, feita inicialmente para SDCD no capítulo 3, pode ser completada em trabalhos futuros. Esta é uma linha de ação vista no momento. Certamente advirão daí necessidades de alterações do SUCO, tendo em vista o novo perfil da comunicação que pode surgir. A automação da manufatura, quando tem suas várias funções executadas de forma integrada e harmônica constitui-se no que é atualmente designado de CIM ("Computer Integrated Manufacturing"). Vê-se que as necessidades de comunicação destas aplicações, tanto em relação a redes locais de comunicação como a protocolos de níveis superiores, são diferentes do que foi visto até então. Primeiro, algumas funções,

como CAD e CAP, em geral não são de tempo real. Segundo, o volume de informação transmitida de cada vez tende a ser alto, do tipo transferência de arquivos(caso tipico de CAD).

Se as aplicações de controle de controle distribuido de processos e CIM forem implementadas baseando-se em redes locais, estas deverão ser bastante gerais. Com a generalidade podem advir problemas de eficiência. Talvez o caminho que a tecnologia vai adotar será o de integrar todas as funções da automação industrial, baseando-se numa rede local de comunicação genérica, porém considerando os SDCD's como um como um componente a mais do sistema distribuido resultante. Estes deverão se ligar à rede através interfaces especiais ("gateways"). Na figura 5.1 tem-se uma possivel configuração dos sistemas integrados de automação industrial. Este é um campo vasto de pesquisa e, como dito, pode consistir num interessante trabalho futuro.

Finalmente, cumpre observar aqui que, no Departamento de Ciência da Computação da UFMG, onde o autor é professor, estão em andamento alguns projetos de pesquisa e desenvolvimento no tema. Um primeiro projeto de pesquisa visa o estudo da comunicação e sincronização entre processos computacionais em ambiente distribuido, do qual este trabalho é parte integrante. Existem também os seguintes projetos de desenvolvimento: a construção de uma rede local de comunicação piloto, para aplicações de controle distribuido de processos em tempo real, incluindo a construção de controladores de comunicação; a especificação e implementação de interfaces entre controlador de comunicação - hospedeiro e/ou sistema operacional; a especificação e implementação de um protocolo de transferência de arquivos; a concepção de um ambiente de aplicações controle distribuido em tempo real, através de simulação; a concepção de um núcleo de programação concorrente para programas distribuidos. A equipe de professores do departamento que atua na área se compõe de três professores. Espera-se para breve a chegada do exterior de mais um professor que está concluindo tese de doutorado na área. Neste ambiente espera-se que o presente trabalho seja continuado, nos aspectos que foram expostos.



WS - Estação de trabalho

SGDBDD - Banco de dados distribuído

SD CP - Sistema distribuído de controle de processos

SD FMS - Sistema distribuído flexível de manufatura

DNC - Comando numérico direto

MF - Máquina ferramenta

Figura 5.1 Possível tendência da automação industrial integrada

BIBLIOGRAFIA

Anderson, G. A. & Jensen, E. D. Computer Interconnnection Structures: Taxonomy, Characteristics, and Examples. In: Tutorial of Distributed Processing, chap.6, IEEE, 1978.

Arabe, J.N.C. & Campos, I.M. Descrição e Simulação de Protocolos de Comunicação de Processos Paralelos Assincronos. In: SEMISH, VIII. Anais do .... Florianópolis, SBC, 1981.

ASEA. Electronic & Computer DS 100. Information 5700 000-B, Edition 2, YLK, jan. 1982.

Bayley Control, Babcock & Wilcox. Bayley Network 90 Catalog. Form CE 93-900B, Wickliffe, USA, 1982.

Bochmann, G . Von. Concepts for Distributed Systems Design. Berlin Heildelberg New York, Springer-Verlag, 1983.

Borsi, L. & Pavlik, E. The Concepts and Structures of Distributed Process Automation Systems. Process Automation, 1980.

Brinch Hansen, Per. Edison - A Multiprocessor Language. Software - Practice and Experience. Vol. II, 1981.

Brinch Hansen, Per. A Key Note Address on Concurrent Programming. Computer, pp: 50-55, maio 1979.

Brinch Hansen, Per. Distributed Processes: A Concurrent Programming Concept. Comm. of the ACM, no. 11, vol. 21, nov. 1978.

Brinch Hansen, Per. The Architecture of Concurrent Programs. Prentice-Hall, Englewood Cliffs,N.J., 1975.

Borsi, L. & Pavlik, E. The Concepts and Structures of Distributed Process Automation Systems. Process Automation, 1980.

Copeliovitch, S. et alii. Automação de Laminadores Continuos de Aços não Planos. In: Congresso Nacional de Automação Industrial - CONAI, 1o.. Anais do ....SEI, SUCESSU/SP, São Paulo, 1983.

EMBRATEL - Serviços RENPAC - Descrição Funcional. Rio de Janeiro, 1984.

Enslow Jr, Philiph H. Multiprocessor Organization - A Survey. In: Tutorial of Distributed Processing, cap 7, IEEE, 1978.

Enslow Jr, Philip H. What is a Distributed Data Processing System?. Computer, Jan 1978.

Fishman, George S. Concepts and Methods in Discrete Event Digital Simulation. New York, John Wiley & Sons,Inc., 1973.

Gentleman, W.M. Message Passing between Sequential Processes: The Replay Primitive and the Administrator Concept. Software - Practice and Experience, vol II, 1981.

Guimarães, C.C. Principios de Sistemas Operacionais. Rio de Janeiro, Ed. Campus, 1980.

Hoare, C.A.R. Communicating Sequential Process. Comm. of the ACM, vol. 21, no. 8, 1978.

Hofmann, W. Problems in the Application of Digital Automation Systems in the Process Industry. Process Automation, no. 2, 1983.

IEEE. IEEE Project 802, Local Networks Standards. Institute of Electric and Electronic Engineers, New York, 1983.

ISO 1983. Information Processing Systems - Open Systems Interconnection - Basic Reference Model. ISO/TC 97/SC 16 N1562-E.

Jensen, E. D. The Honeywell Experimental Distributed Poocessor - An Overview. Computer pp: 28-37, jan. 1978.

Jones, A.K. & Schwarz, P. Experience Using Multiprocessors - A Status Report. Computing Surveys, no. 2, vol. 12, 1980.

Kahne, S. & Lefkowitz, I. & Rose, C. Automatic Control by Distributed Intelligence. Electronics, 1982.

Kleinrock, Leonard. Queueing Systems. Volume 1: Theory. New York, Wiley, 1975.

Kleinrock, Leonard. Queueing Systems. Volume 2: Computer Applications. New York, Wiley, 1976.

Kobayashi, Hisashi. Modeling and Analisys: An Introduction to System Performance Evaluation Methodology. Reading, Addison- Wesley Publishing Company,Inc., 1978.

Lages, Newton A. C. Comunicação em Redes Locais de Computadores para Controle Distribuido de Processos. Campinas-SP, UNICAMP 1981 (Tese de Doutoramento).

Lauer, J.C. & Needham, R.M. On the Duality of Operating Systems Structures. In: International Symposium on Operanting Systems, 2nd, Proc of the ... IRIA, 1978.

Leeds & Northrup Company. MAX1 Distributed Control - Functional Description. Ref 27720l, Rev. C., North Vales, USA, 1982.

LeLann, G. Motivations, objectives and characterization of distributed systems. In: Distributed Systems - Architecture and Implementation. Ed.: Lampson & Paul & Siegert. Springer-Verlag, Berlin Heildelberg New York, 1981.

Liebowitz, B.H. Introduction to Chap. 1. In: Tutorial on Distributed Processing, chap. 1, IEEE 1978. Edited by Liebowitz B.H. & Carson J.H.

Liu, M. T. Distributed Loop Computer Networks. In: Advances in Computers, vol 17, pp 163-221. Academic Press, Inc., 1978.

Mendes, M.J. As Novas Tecnologias da Automação da Manufatura. Rio de Janeiro, Jornal Data News, Suplemento Especial de Automação Industrial, 29/05/84.

Manning, E. & Livesey, N.J. & Tokuda, H. Interprocess Communication in Distributed Systems: one View. Information Processing. S.H. Lavingston (ed.). North Holland Pub. Co., IFIP 1980.

Metcalfe, R. M. & Boggs, D. R. Ethernet: Distributed packet switching for local computer networks. Comm. ACM, vol 19, no. 7 (July), 1976.

Myers, Glenford J. Composite/Structured Design. New York. Van Nostrand Reinhold Company, 1978.

Myers, W. Towards a Local Network Standard. IEEE Micro, vol2, no. 3 (Aug), 1982.

Needham, R. M. Systems Aspects of the Cambridge Ring. In: Symposium on Operating Systems Principles, 7th. Proc of the... ACM-SIGOPS, Pacific Grow-CA, 1979.

Castilho, N.A. & Nogueira, J.M.S. Aspectos do Controle Distribuido de Processos. Revista Controle & Instrumentação. Artigo convidado, abril, 1985.

Nogueira, J.M.S. Modelagem de Procedimentos e Monitores usando a Metodologia do NUCA: Contribuições. Belo Horizonte, DCC-UFMG, RT 002/84, 1984.

Nogueira, J.M.S. Simulação de Redes de Filas no Auxilio a Projeto, Especificação e Avaliação de SDCD. Belo Horizonte, DCC-UFMG, RT 006/84, 1984.

Nogueira, J.M.S. & Ruggiero, W.V. & Mendes, M.J. Redes Locais para Controle de Processos em Tempo Real: Comunicação e Arquitetura. In: Convencion Informática Latina - CIL/83, Anais da .... Barcelona, junho de 1983.

Nogueira, J.M.S. & Mendes, M.J. & Ruggiero, W.V. Comunicação em Redes Locais de Computadores para Controle Distribuido de Processos em Tempo Real. In: Congresso Brasileiro de Automática, 4o., Anais do.... Campinas-SP, SBA, 1982.

Nogueira, J.M.S. & Lages, N.A.C. Aspectos de Comunicação e Sincronização num Ambiente de Controle Distribuido. In: Congresso Nacional de Automação Industrial - CONAI, 1o.. Anais do... SEI, SUCESSU/SP, São Paulo, 1983.

Paula Filho, W.P. & Campos, I.M. Aplicação de um Núcleo de Concepção Arquitetônica à Automação de Projetos de Sistemas Digitais. In: SEMISH, VIII. Anais do ....Florianópolis, SBC, 1981.

Penney, B. K. & Baghdadi, A. A. Survey of computer communications loop networks: Part 1. Computer Communications, vol 2, no. 4, 1979.

Penney, B. K. & Baghdadi, A. A. Survey of computer communications loop networks: Part 2. Computer Communications, vol 2, no. 5, 1979.

Rosemberg, R. Closing in Open Systems. Electronics, May 31, 1984.

Ruggiero, W.V. & Melnikoff, S.S.S. & Shimizu, E.Y. DAM: A Distributed Architecture Machine. In: International Symposium of Mini and MicroComputers, 15o.. Anais do ... Mexico, abril, 1981.

Sauer, Charles H. & Chandy, K. Mani. Computer Systems Performance Modeling. Englewood Cliffs, Prentice-Hall,Inc.,

1981.

Schoeffler, J.D. Distributed Computer Systems for Industrial Process Control. Computer, Feb. 1984.

SEI - Secretaria Especial de Informática. Documento Básico dos SDCD's. Brasília, 1982.

Stallings, W. Local Networks. Computing Surveys, vol 16, no. 1, March 1984.

Steusloff, H.V. Structures of Automatic Control Systems and the Consequences for Programmming Languages. Process Automation, no. 1, pp: 3-11, 1979.

Steusloff, H.V. Advanced Real-Time Languages for Distributed Industrial Process Control. Computer, Feb. 1984.

Tanenbaum, A. S. Computer Networks. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1981.

Tozzi, C.L. & Andrade Neto, M.L. & Melnikoff, S.S.S. Introdução aos Sistemas de Tempo Real para Controle de Processos. In: Congresso Nacional de Automação Industrial - CONAI, 1o.. Anais do... SEI, SUCESSU/SP, São Paulo, 1983.

Watson, R.W. Distributed system architecture model. In: Distributed Systems - Architecture and Implementation. Ed.: Lampson & Paul & Siegert, Spring-Verlag, Berlin Heildelberg New York, 1981.

Weidlich, S. & Prutz, G. Selection Criteria for the Use of Digital Distributed Control Systems. Process Automation - PA, 1982.

Zimmermann, H. OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection. IEEE Trans. Commun. Vol COM 28, pp 425-432, April 1980.

Zuchi, W. & Ruggiero, W.V. Protocolos para Serviços de Tempo Real em Rede de Computadores. EPUSP, Relatório Técnico, 1982.