SISDI-OSI: SISTEMA DIDÁTICO PARA O MODELO OSI

Manuel de Jesus Mendes Edmundo Roberto Mauro Madeira
Flávio Morais de Assis Silva
Elza Kiyomi Shimabukuro Garcia
Lisiane Maria Bannwart Ambiel
Luiz Otávio Merlin Miranda
María Inés Valderrama Restovíc Milton T. Sakamoto
Célio Toshihiro Fujito
UNICAMP
e-mails: mendes@dca.fee.unicamp.br edmundo@dcc.unicamp.br

31 de março de 1993

Sumário

Este trabalho apresenta o Sistema Didático SISDI-OSI em desenvolvimento pelo Grupo de Redes da Faculdade de Engenharia Elétrica e do Departamento de Ciência da Computação da UNICAMP. O Sistema é composto por diversos protocolos do Sistema de Aplicação (camadas de sessão, apresentação e aplicação) do Modelo de Referência OSI/ISO (*Open Systems Interconnection / International Organization for Standardization*). Este artigo analisa aspectos de implementação dos protocolos e a integração dos mesmos, além de comentar como está sendo realizada esta implementação no ambiente UNIX. Uma implementação do compilador ASN.1 também é apresentada.

Abstract

This work presents the Didactic System SISDI-OSI, which is being developed by the Network Group of the Faculty of Electrical Engineering and the Department of Computer Science at UNICAMP. The System is composed by several upper-layer protocols (session, presentation and application protocols) of the Reference Model for Open Systems Interconnection of the International Organization for Standardization (OSI/ISO). This paper analyzes some protocol implementation issues and the integration of these protocols, besides commenting how it is being done in UNIX. An implementation of the ASN.1 Compiler is also presented.

1 INTRODUÇÃO

A integração de equipamentos isolados em redes de computadores, sejam de longa distância ou locais, gerou a necessidade de desenvolver arquiteturas de Sistemas de Comunicação. Um modelo de arquitetura muito utilizado é o Modelo de Referência OSI/ISO, que é composto por sete camadas. As quatro inferioras (física, enlace, rede e transporte) são responsáveis pelo transporte das mensagens entre os sistemas finais, enquanto que as três superioras (sessão, apresentação e aplicação) estão relacionadas com as aplicações.

Cada camada tem por finalidade resolver alguns problemas relacionados, oferecendo para a camada superiora estes serviços. Por exemplo, a camada de rede trata do roteamento de mensagens e a camada de transporte detecta e recupera erros de mensagens entre os sistemas finais. Para cada camada, a ISO define diversos tipos ou classes de protocolos para resolverem os problemas específicos desta camada em função da complexidades dos mesmos. Por exemplo, para a camada de rede a ISO define dois protocolos: com conexão (mais complexo e confiável) e sem conexão (mais simples e menos confiável); e para a camada de transporte, cinco classes, em função da confiabilidade das sub-redes existentes e de outros parâmetros. Note que, dependendo do protocolo de rede, deve ser escolhida uma classe diferente de protocolo de transporte. O SISDI-OSI implementa alguns protocolos da ISO para as três camadas superioras.

Para a camada de aplicação, que provê serviços de forma compreensível para os Processos de Aplicação (APs), a ISO definiu diversos protocolos para atender aos vários tipos de uso, como por exemplo:

MMS (Message Manufacturing Specification) para o ambiente industrial;

RDA (Remote Database Access) para acesso às Bases de Dados Remotas;

TP (Transaction Processing) para o Processamento de Transações Distribuídas;

DS (Directory Service) para prover os serviços de Diretório;

NM (Network Management) para prover os serviços de gerenciamento de redes;

ROSE (Remote Operations Service Element) para prover os serviços de operações remotas;

ACSE (Association Control Service Element) para gerenciar as associações de aplicação.

O SISDI-OSI surgiu a partir do SISDI-MAP - Sistema Didático do Protocolo e da Interface de Aplicação MMS do MAP (*Manufacturing Automation Protocol*), desenvolvido também na UNICAMP [2]. O SISDI-MAP foi implementado

para microcomputadores compatíveis com o PC, e é executado num ambiente DOS com a adição de um núcleo de tempo real. O SISDI-OSI, por sua vez, está sendo implementado nas estações SUN da UNICAMP com o ambiente UNIX. O SISDI-MAP possui os protocolos MMS e ACSE definidos pela ISO e adotados pelo Projeto MAP, enquanto que o SISDI-OSI possui, além destes protocolos, outros definidos pela ISO: RDA, TP, DS, NM, ROSE, CCR (*Commitment, Concurrency and Recovery*), de apresentação e de sessão. No SISDI-OSI apenas subconjuntos dos serviços oferecidos pelos diversos protocolos são implementados. Os dois sistemas possuem a Interface de Aplicação (API) para o protocolo MMS.

Os objetivos do SISDI-OSI são o de obter experiência de implementação de protocolos e o de ter um sistema didático que pode ser utilizado em aulas práticas sobre protocolos para comunicação.

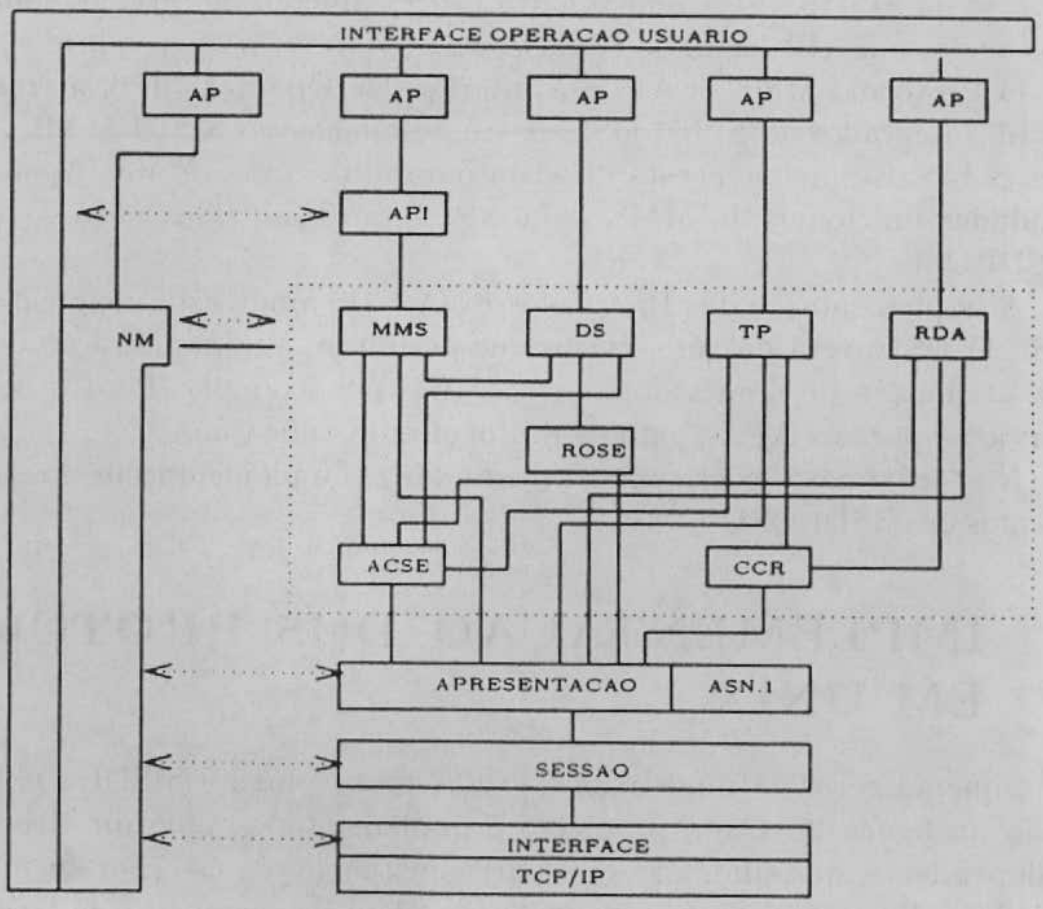

Figura 1: Modelo do SISDI-OSI

O modelo conceitual do SISDI-OSI é apresentado na figura 1 e apresenta os seguintes blocos funcionais:

API executa as funções da Interface de Aplicação para o MMS;

NM, MMS, RDA, TP, DS, ROSE, ACSE, CCR, Apresentação, Sessão executam as funções dos respectivos provedores de serviços;

Interface de Operação de Usuário permite ao usuário do SISDI-OSI criar uma chamada de função da API ou dos diversos protocolos de aplicação utilizando um AP para este fim (opção A), ou executar um AP já existente (opção B). Além disso, fornece ao usuário informações sobre as mensagens enviadas/recebidas para/da rede quando tratadas nos diversos processos;

ASN.1 executa as funções do compilador ASN.1;

Interface realiza o mapeamento entre o mundo OSI/ISO (sessão) e o mundo TCP/IP (*Transmission Control Protocol/Internet Protocol*).

Nesta figura, as linhas com setas indicam o tráfego de informação para o usuário do SISDI-OSI acompanhar o processamento de serviços solicitados aos protocolos e à API, além do envio e recepção de mensagens entre si.

O Protocolo MMS, a API e a Interface de Operação de Usuário ainda não foram integrados ao SISDI-OSI, existindo apenas no SISDI-MAP. A migração destes blocos funcionais está prevista para uma etapa futura. Apenas algumas unidades funcionais do MMS e da API serão traduzidas para o ambiente do SISDI-OSI.

A implementação dos Protocolos RDA e DS ainda está incipiente no SISDI-OSI. O DS deverá prover serviços que permitam a administração de endereços e a atribuição de nomes dos objetos OSI (por exemplo, Pontos de Acesso de Serviços) para os APs e para os protocolos de aplicação.

Nas próximas seções serão comentadas as implementações dos demais protocolos do SISDI-OSI.

2 IMPLEMENTAÇÃO DOS PROTOCOLOS EM UNIX

O esquema geral de implementação dos protocolos no SISDI- OSI é apresentado na figura 2. Cada protocolo é implementado como um processo UNIX independente, que simula as suas várias instâncias de uso (por exemplo, usando tabelas). Para receber mensagens, que podem ser primitivas de serviços, PDUs ou primitivas de controle, há uma única fila, de endereço conhecido pelos que necessitam enviar mensagens ao protocolo. Cada processo acessa ainda uma área de memória compartilhada por todos os processos, que contém um *buffer*, usado para armazenar primitivas e PDUs.

Cada componente MACF (*Multiple Association Control Function*), que controla atividades relacionadas em mais de uma associação, e SACF (*Single Association Control Function*), que desempenha as funções de controle das atividades realizadas por diversos ASEs em uma única associação, são implementados

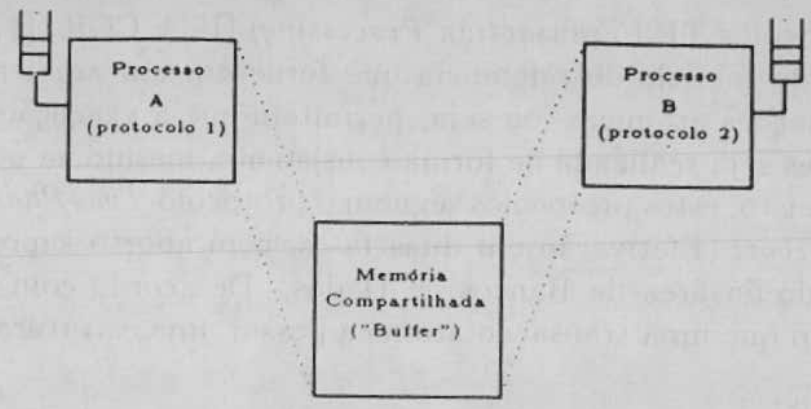

Figura 2: Esquema geral de implementação dos ASEs

também desta maneira. Um ASE (*Application Service Element*) corresponde a um protocolo da camada de Aplicação ou parte dele.

As filas e a área de memória compartilhada são implementadas utilizando, respectivamente, o mecanismo de mensagens e de compartilhamento de memória do UNIX System V [3], para comunicação inter-processos. Quando um processo deseja enviar uma mensagem a um outro processo, ele aloca uma área no buffer, escreve nesta área o conteúdo da mensagem, e coloca uma mensagem na fila do processo destino, cujo conteúdo é mostrado na figura 3.

id_invoc	componente	tipo_info	apontr_info

Figura 3: Formato da mensagem colocada nas filas

Esta mensagem possui um campo que identifica a instância do protocolo à qual a mensagem se destina (campo *id_invoc*); um que indica a qual protocolo a mensagem está relacionada (campo *componente* - por exemplo, se a mensagem representa uma primitiva do TP, este campo terá o valor TP); um outro campo, que indica se a mensagem contém dados de uma PDU ou de uma primitiva (campo *tipo_info*); e um apontador para o local do buffer onde os dados realmente estão armazenados (*apontr_info*).

É utilizado ainda o mecanismo de semáforos, também do UNIX System V [3], para o controle de acesso concorrente ao *buffer*.

3 O TP E O CCR

Os protocolos TP (*Transaction Processing*) [19] e CCR ([17] e [18]) são os protocolos do modelo de referência que fornecem um ambiente para a execução de transações atômicas, ou seja, permitem que a execução de um conjunto de operações seja realizada de forma consistente, mesmo na ocorrência de falhas.

Para isto, estes protocolos seguem o protocolo *Two-Phase Commit with Presumed Abort* (Efetivação em duas fases, com aborto suposto) [4], já bastante conhecido da área de Bancos de Dados. De acordo com ele, o TP e o CCR assumem que uma transação atômica possui uma estrutura de árvore, como na figura 4:

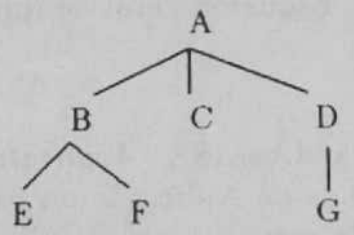

Figura 4: Árvore de Transação Atomica

Cada nó da árvore (marcados com as letras) corresponde a uma AEI (*Application Entity Invocation*), que representa uma ocasião de uso de uma AE (*Application Entity*). As AEs são agrupamentos de protocolos e funções de controle da camada de Aplicação, usados por um AP para um determinado objetivo de comunicação. Por exemplo, o conjunto dos protocolos TP, RDA, CCR e ACSE e funções de controle formam uma AE para acesso a bancos de dados remotos. Uma aresta liga dois nós somente se as AEIs que eles representam precisam-se comunicar para a execução da transação atômica. Esta árvore é chamada de *Árvore da Transação Atômica*.

O CCR é o protocolo que controla as interações em cada ramo (aresta) desta árvore, ou seja, a comunicação entre duas AEIs. O TP controla as interações na árvore como um todo, levando em conta os resultados nas várias associações entre as AEIs.

As implementações destes protocolos foram feitas usando ESTELLE [16], que é uma linguagem de especificação formal da ISO, com a inserção de funções auxiliares em C. Em [19] há uma especificação do TP e de parte de seu SACF e MACF nesta linguagem. Porém, para a implementação, adaptações tiveram que ser feitas nesta especificação. Para o caso do CCR, a implementação foi feita com base apenas na descrição do protocolo em linguagem natural ([17, 18]).

A estrutura da implementação destes protocolos segue o modelo geral, descrito na seção anterior, ou seja, ao CCR corresponde um processo e ao TP, três: um para o TP-ASE (componente que representa o protocolo TP em si) e outros

dois para suportarem o SACF e MACF necessários. Só será implementado o suporte do TP a operações atômicas em banco de dados remotos, ou seja, as interações entre o TP, o RDA e os protocolos de cujos serviços necessitem: ACSE e CCR.

A estrutura geral em ESTELLE da implementação de cada um dos componentes (figura 5) é composta de um módulo INTERFACE, que faz a compatibilização do módulo implementado nesta linguagem com os demais processos do sistema, transformando mensagens da fila externa do processo em mensagens ESTELLE e vice-versa; de várias instâncias do módulo principal da especificação em ESTELLE (representadas na figura pelas caixas entituladas MÓDULO PRINCIPAL), uma para cada instância de uso do protocolo; e de um módulo ESPECIFICAÇÃO, que engloba todos os outros e trata da criação e destruição das instancias dos MÓDULOS PRINCIPAIS. Os MÓDULOS PRINCIPAIS correspondem aos códigos para o TP, o CCR, o SACF e o MACF.

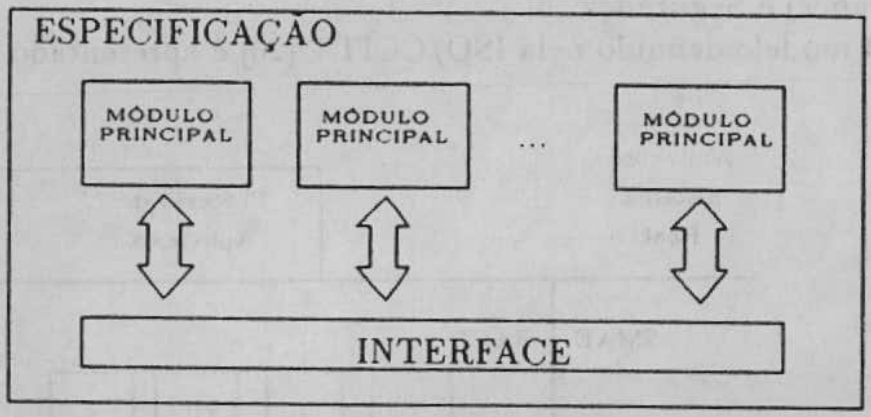

Figura 5: Estrutura Geral dos Módulos em ESTELLE

Não estão sendo implementadas todas as unidades funcionais do TP, apenas as unidades *Kernel*, *Commit*, *Shared Control* e *Chained Transactions*. O ambiente utilizado foi o EWS [1], que gera código em C a partir da especificação em ESTELLE.

A linguagem ESTELLE fornece uma estrutura adequada para a implementação dos protocolos com fins didáticos. A implementação nesta linguagem reflete mais o que está escrito nas especificações, facilitando a compreensão tanto da implementação quanto da própria especificação. Possibilidade também o uso de ferramentas como simuladores.

Embora as implementações estejam sendo feitas como especificações ESTELLE separadas, elas são estruturadas de modo que possam ser unidas facilmente para gerar uma única especificação global. Pelas facilidades fornecidas pela linguagem, não é dificil compatibilizar o protocolo implementado em ESTELLE com os demais processos do sistema, desenvolvidos em C. Esta união permitiria o estudo, através de ferramentas de análise, do comportamento desta

especificação mais genérica.

4 GERÊNCIA DE SISTEMAS

Entende-se por gerência a atividade de controlar e monitorar os recursos de um sistema.

No ambiente de sistemas abertos existem várias atividades gerenciáveis: capacidade de processamento, capacidade de armazenamento e capacidade de comunicação. É importante observar, porém, que o trabalho de padronização realizado sobre Gerência de Sistemas envolve apenas os recursos relacionados à comunicação entre os nós de uma rede.

Os recursos a serem gerenciados são denominados de objetos gerenciados.

Estas atividades são ainda agrupadas em 5 funções [10], designadas a seguir de FCAPS: Falhas, Configuração, Tarifação (*Accounting*), Desempenho (*Performance*) e Segurança.

O modelo definido pela ISO/CCITT [20] é apresentado na figura 6.

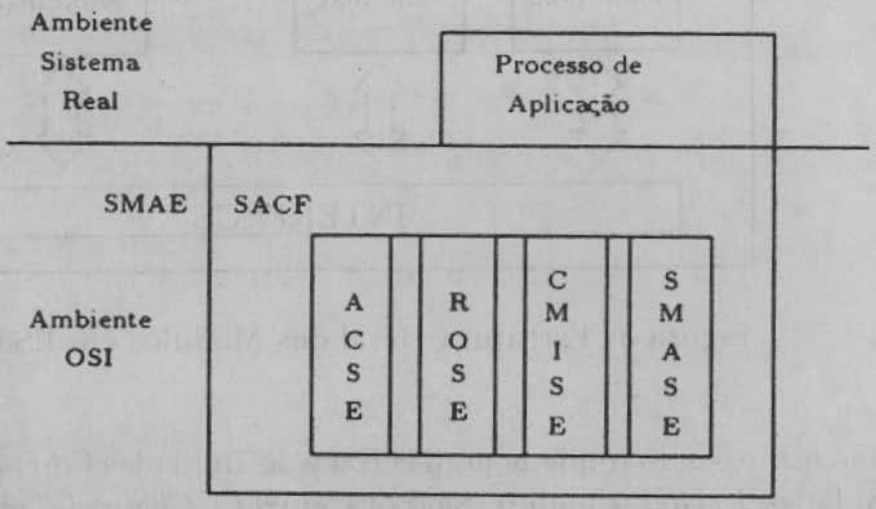

Figura 6: Modelo definido pela ISO/CCITT para Gerência de Rede

No modelo são ainda definidos dois conceitos: agente e gerente. O gerente é responsável pelas operações de gerenciamento de uma rede. O agente é responsável pela execução das operações sobre os objetos gerenciados, a pedido do gerente e pelo envio para o gerente de notificações de eventos ocorridos nos objetos gerenciados.

Este modelo utiliza os serviços oferecidos pelos ASEs ACSE e ROSE.

No projeto SISDI-OSI, a implementação de um sistema de Gerência de Sistema visa à criação de um sistema didático de visualização de eventos ocorridos na rede, bem como da interação do operador com o sistema, gerando a ocorrência de certos eventos.

Figura 7: Troca de Informação entre Gerente e Agente

O modelo do Sistema Didático de Gerência de Redes (SDGR) é apresentado nas figuras 8 e 9.

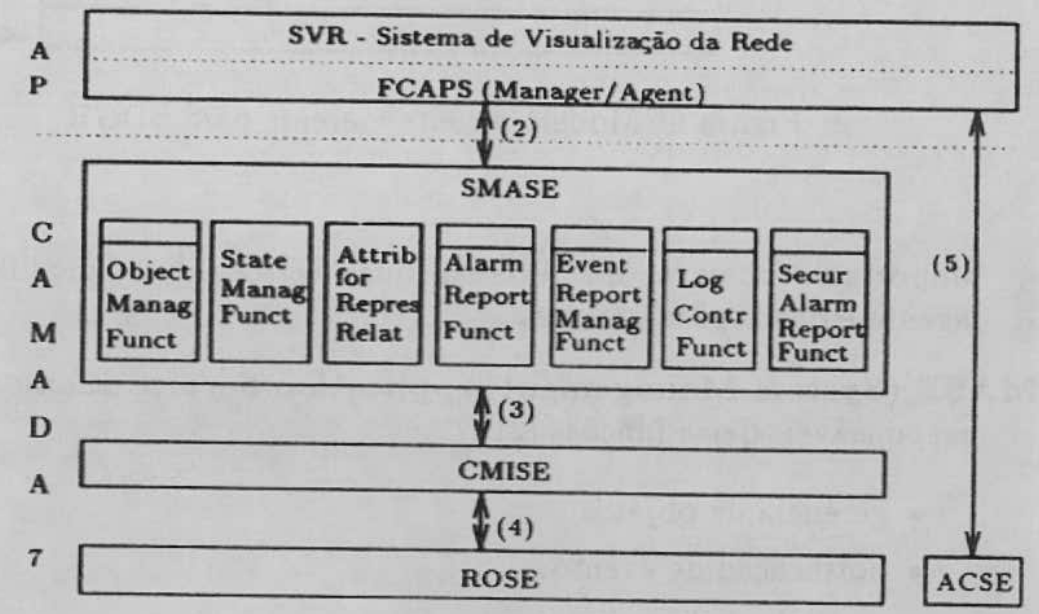

Figura 8: Modelo do SDGR

O SDGR é composto dos seguintes módulos:

SVR – Sistema de Visualização da Rede módulo responsável pela interface Homem-Máquina, permitindo ao operador do sistema realizar as funções de gerência de rede para visualizar os eventos ocorridos.

FCAPS módulo responsável pelo gerenciamento das funções FCAPS, analisando quais delas farão parte da operação do gerenciamento corrente. É

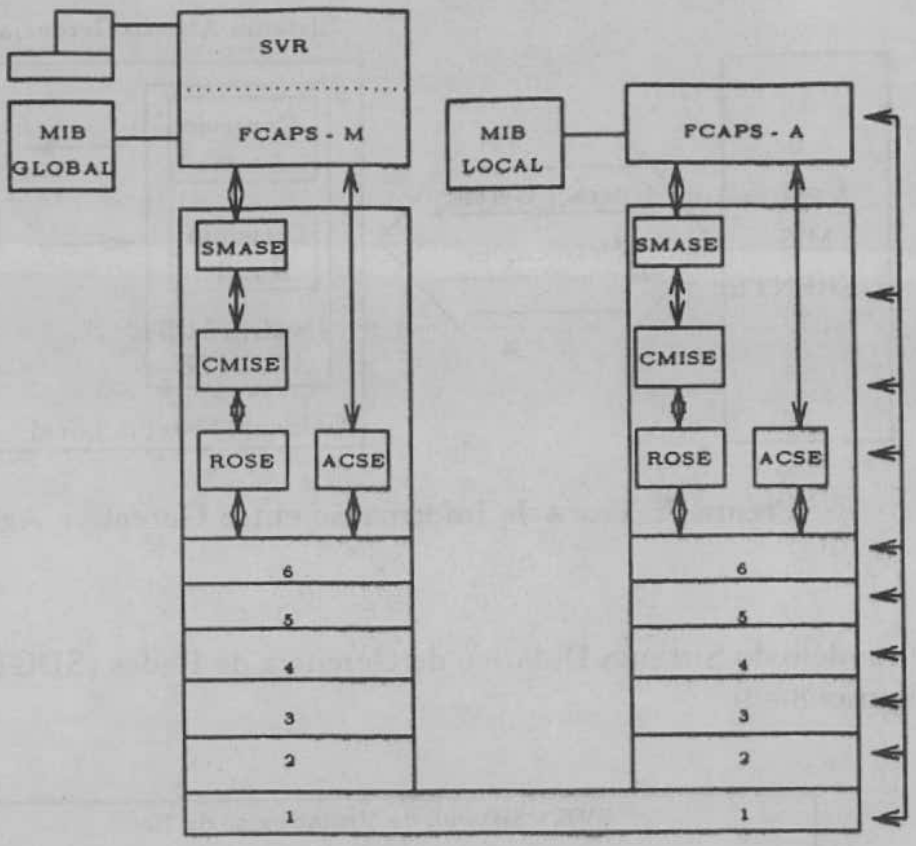

Figura 9: Modelo Agente-Gerente para SDGR

importante observar que existem duas versões deste módulo: um para agente e outro para gerente.

SMASE (System Management Application Service Element) módulos responsáveis pelas funções [21]:

- gerência de objetos
- notificação de eventos
- controle de logs
- notificação de alarmes de segurança
- gerência de estados

CMISE (Common Management Information Service Element) módulo responsável pela implementação do protocolo de transferência de informações de gerenciamento ([7], [8]).

ACSE e

ROSE

Para a execução das operações de gerenciamento, é necessária a existência no agente de uma interface direta, para atuação sobre os objetos gerenciados, entre o módulo FCAPS-A e a cada uma das camadas do modelo OSI.

No SDGR a ser implementado existirão duas Bases de Dados:

MIB global Base de Dados contendo a configuração de Objetos Gerenciados

MIB local Base de Dados contendo informações sobre a identificação local do objeto

5 A CAMADA DE APRESENTAÇÃO

A Camada de Apresentaçaõ é responsável pela representação dos dados em trânsito entre Sistemas Abertos, mantendo o seu significado para que as Camadas de Aplicação, local e remota, sejam capazes de comunicar entre si. É responsabilidade das entidades de aplicação determinar o conjunto de sintaxes abstratas que serão utilizadas e informar a entidade de apresentação sobre o seu acordo. É responsabilidade das entidades de apresentação selecionar sintaxes de transferência mutuamente aceitáveis e capazes de representar os valores dos dados. À combinação Sintaxe Abstrata/Sintaxe de Transferência é dado o nome de Contexto de Apresentação. Ao menos um Contexto de Apresentação deve ser conhecido pela Camada de Apresentação (Contexto *Default*) [6, 5].

Na fase inicial do SISDI-OSI, a Camada de Apresentação foi implementada somente com a Unidade Funcional *Kernel* para executar em ambiente DOS, em máquinas IBM-PC ou compatíveis. Foi utilizado o software EPOS (*Engineering Project Oriented System*) [9] para especificação do projeto e geração semi-automática do código em linguagem C. Os procedimentos relacionados ao *Kernel* são suficientes para estabelecimento de uma conexão de apresentação, transferência de dados e liberação da conexão.

Na fase atual, este código está sendo adaptado para o ambiente UNIX e para novas estruturas de dados tendo por base o uso de ASN.1 (*Abstract Syntax Notation One*) e BER (*Basic Encoding Rules*) que serão vistos na próxima seção. Além disso serão incorporadas as rotinas de codificação e decodificação das PDUs como parte das funções da Camada de Apresentação. O uso do EPOS nesta fase se restringirá à especificação, não mais sendo utilizado para a geração de código uma vez que já se mostrou restrito no tratamento de dados.

O próximo passo após esta adaptação ao ambiente UNIX e às novas estruturas de dados será a incorporação dos serviços referentes ao Gerenciamento de Atividades e Sincronização (Unidade Funcional de Sessão).

6 ASPECTOS DA IMPLEMENTAÇÃO DE UM COMPILADOR ASN.1

A ASN.1 [13, 14] é uma notação para sintaxe abstrata que descreve, de uma maneira independente, os tipos de dados que formam parte das PDUs dos protocolos de aplicação; relacionada às sintaxes abstratas, existe sintaxes de transferência que, por sua vez, definem um conjunto de regras para representação física sem ambigüidades dos valores dos tipos de dados. Não existe uma representação computacional de ASN.1, portanto usa-se a linguagem C para representação, uma vez que é a utilizada na implementação dos protocolos do sistema.

A construção de um compilador que efetue esta transformação foi necessária uma vez que os tamanhos das PDUs podem ser muito grandes, o que dificulta a realização desta operação manualmente. Outro ponto importante a considerar é a padronização das estruturas de dados utilizadas nas implementações do sistema. Um segundo objetivo do compilador é gerar as rotinas de codificação e decodificação das PDUs das camadas de Aplicação e Apresentação, as quais serão chamadas por esta última. Estas rotinas realizam a conversão dos valores dos dados das PDUs da forma como é armazenada localmente para a forma de codificação especificada em BER e vice-versa [15].

Basicamente, o compilador ASN.1 [22] foi implementado utilizando as ferramentas LEX (*Lexical Analizer Generator*) e YACC (*Yet Another Compiler Compiler*), como também a linguagem C do próprio sistema operacional. O compilador foi definido em dois passos, ambos determinados pelas características de ASN.1 que diferenciam de C. Por ASN.1 ser uma linguagem que determina que as declarações de seus tipos sejam feitas na forma *top-down* e a organização das declarações de C serem feitas na forma —em botton-up, o primeiro passo do compilador realiza uma inversão da especificação ASN.1 para adaptá-la à forma utilizada em C. Uma vez que o código de entrada foi preparado, um segundo passo realiza todas as fases de compilação: análise léxica, análise sintática e análise semântica, para obter como resultado final a geração de um código equivalente em C.

Para efetuar a transformação entre ambas as linguagens foi atribuído, para cada tipo definido em ASN.1, um tipo de C, seja um tipo padrão ou uma estrutura de dados definida com as características do tipo que se deseja representar. Por exemplo: os tipos INTEGER, BOOLEAN, REAL podem ser traduzidos diretamente nos tipos int e float de C, mas os tipos BITSTRING e EXTERNAL precisam de estruturas definidas especialmente para representá-los. Adicionalmente ASN.1 associa certas características e propriedades para seus tipos, as quais são inexistentes em C, como as "tags". Estas características serão representadas na tabela de símbolos gerada na compilação e que é utilizada também pela codificação e decodificação dos dados.

Uma facilidade de ASN.1 a ser considerada fora do contexto de uma com-

pilação tradicional são as Macros. Uma macro em ASN.1 difere de uma macro em uma linguagem comum, pois esta última só substitui uma parte de código por outra. Em ASN.1 pode-se definir um *novo tipo de dado*, especificando novas regras gramaticais, através da qual as regras gramaticais que definem ASN.1 são incrementadas. Devido à existência de macros, muito utilizadas, por exemplo, no protocolo ROSE, este conjunto de macros foi incluído na própria gramática da linguagem ASN.1, como se este conjunto fizesse parte de seus tipos simples.

Finalmente, as rotinas de codificação e decodificação das PDUs realizam chamadas a funções de codificação e decodificação para cada tipo de dado básico definido em ASN.1, que se encontram em duas bibliotecas auxiliares; este fato constitui a parte dinâmica da utilização dos produtos da compilação. Portanto, cada vez que se deseja codificar uma PDU, só é necessário compilá-la uma vez, realizada *off-line*, para se obter as rotinas específicas e o código C, embora as rotinas possam ser requeridas quantas vezes seja necessário.

7 A CAMADA DE SESSÃO

A Camada de Sessão tem como função estabelecer uma conexão com o usuário-par, permitir a troca de dados de forma sincronizada entre os usuários de Sessão e liberar a conexão de forma ordenada. Em contraste com os níveis mais baixos da rede, que geralmente assumem que devem executar mecanismos bem elaborados para fornecer um serviço confiável para os níveis superiores, o nível de sessão assume estar recebendo unidades de dados confiáveis da rede, assim ele usa poucos procedimentos de controle de erros [11, 12].

Na fase inicial do SISDI-OSI, a Camada de Sessão foi implementada com as Unidades Funcionais *Kernel* e *Full-Duplex*, também para o ambiente DOS. Assim como a Camada de Apresentação, a Sessão também utilizou o software EPOS [9] para a especificação formal do protocolo e a geração semi-automática de código em linguagem C.

Este código passa hoje por um processo de adaptação ao novo ambiente de trabalho e às novas estruturas de dados que não estavam de acordo com os padrões OSI.

Os serviços a serem incorporados correspondem às Unidades Funcionais Liberação Ordenada, *Half-Duplex*, Sincronização Maior e Menor e Gerenciamento de Atividades. Com isso alguns conceitos ainda não abordados passarão a fazer parte do protocolo, como:

Token atributo de uma conexão de sessão, cuja posse dá ao usuário o direito de usar exclusivamente um determinado serviço;

Pontos de Sincronização utilizados para coordenar a troca de dados, de forma a permitir uma recuperação durante a transmissão de dados. A sincronização maior define início e fim de unidades de diálogo e a sincronização menor permite a estruturação do diálogo;

Atividade formada por unidades de diálogo (conjunto de dados logicamente relacionados).

8 INTERFACE CAMADA DE SESSÃO/TCP

No SISDI-OSI apenas a interface de serviços para a classe 4(TP4) será oferecida para a camada de sessão. Isto deve-se ao fato desta ser a mais completa e apresentar uma boa qualidade de serviço, independente dos recursos/restrições da camada de rede. Esta característica favorece a adaptação do SISDI-OSI em diferentes plataformas. O TCP tem uma grande semelhança com o TP4.

Para o SISDI-OSI, dentre os recursos oferecidos pelo sistema operacional UNIX, um destes será amplamente explorado pela interface TP4 com o objetivo de prover comunicação entre processos: o *socket* [24]. O *socket* provê um canal de comunicação entre dois processos, sendo este canal estabelecido em tempo de execução e nao é necessário que tais processos tenham um "parentesco" (relação pai/filho criado via *fork*).

Uma vez iniciado, o *socket* pode ser utilizado para receber uma indicação de conexão vinda de outro processo ou requisitar uma conexão a outro processo. Uma vez estabelecida a conexão, a troca de dados é *full-duplex*. Nesta implementação, cada conexão é tratada por uma instância do processo pai, onde cada uma é criada na conexão e finalizada na desconexão. Isto torna flexível o sistema de forma a permitir a conexão simultânea com diferentes entidades de transporte ou comunicar com uma mesma entidade via duas ou mais conexões.

Existe ainda o problema de endereçamento. A entidade de transporte deve ser conhecida/acessada por qualquer entidade par. A nível de rede, cada estação é endereçada por um inteiro longo (4 bytes) ou por um nome lógico. A identificação das estações é obtida via chamadas ao sistema que por sua vez consulta o servidor de endereço da rede (NIS - *Network Information Service*) [23].

Uma vez acessada a estação desejada, deve existir um meio de identificar/acessar a interface TP4. Esta identificação se fará mediante a utilização de uma porta (*Port*) pré-definida. A esta estará associado um *socket* em estado de espera de requisição de conexão. Na inicialização do *socket*, o sistema aloca e lhe associa uma porta. Esta alocação não está sob o controle da interface TP4. Dessa forma, a cada execução, não necessariamente a mesma porta será selecionada. A interface TP4 remota deve ser notificada de alguma forma toda vez que a interface TP4 local for iniciada. Para contornar esse problema, a interface TP4 tem a opção de selecionar uma porta vaga dentro de um conjunto reservado aos processos de aplicação. Isto torna possível que todas as interface TP4 utilizem a mesma porta nas suas respectivas estações, o que faz com que os acessos às portas das interfaces TP4 remotas possam ser feitos sem a prévia notificação.

9 CONCLUSÃO

O SISDI-OSI permite aos seus usuários ter uma compreensão maior dos protocolos implementados e da integração dos mesmos, além de incentivar a utilização do Modelo OSI/ISO através do conceito de Sistema Aberto. No SISDI-OSI poder-se-á trocar, adicionar ou retirar protocolos sem alterar a execução do restante do sistema. No SISDI-OSI entende-se melhor as interfaces entre protocolos e entre protocolos e APs, interfaces estas que fogem ao escopo das especificações da ISO.

As implementacpes do SISDI-OSI têm requerido um estudo detalhado das especificações da ISO/CCITT. Isto tem feito com que os próprios conceitos presentes nestas especificações sejam analisados, o que tem resultado na aquisição de um conhecimento mais aprofundado do modelo RM-OSI/ISO e de suas capacidades e deficiências.

O SISDI-OSI poderá também ser utilizado como uma plataforma de trabalho para aplicações em sistemas distribuídos sempre que o aspecto didático (visualização do que está ocorrendo em cada parte do sistema) for fundamental.

AGRADECIMENTOS

Os autores gostariam de agradecer aos que trabalharam na implementação do SISDI-MAP por terem transmitido a nós suas experiências.

Referências

[1] *EWS User's Manual - Esprit Project 265 - SEDOS Estelle Demonstrator*, Junho 1989.

[2] Edmundo Roberto Mauro Madeira et al. Antenor Paglioni Junior, Durval Carvalho Ávila Jacintho. SISDI-MAP: Sistema Didático do Protocolo e da Interface de Aplicação MMS do MAP. In *Seminário Franco-Brasileiro em Sistemas Informáticos Distribuídos*, Florianópolis, SC, Setembro 1989.

[3] Maurice J. Bach. *The Design of the UNIX Operating System*. Prentice-Hall Software Series. Prentice-Hall, Inc., EUA, 1986.

[4] B. Lindsay C. Mohan. Efficient Commit Protocols for the Tree of Processes Model of Distributed Transactions. Technical report, IBM Research Laboratory, San Jose, EUA, 1983.

[5] CCITT. Presentation Protocol Specification for Open Systems Interconnection for CCITT Applications - Recommendation X.226, 1988.

[6] CCITT. Presentation Service Definition for Open Systems Interconnection for CCITT Applications - Recommendation X.216, 1988.

[7] CCITT. Draft Recommendation X.710 - Common Management Information Service Definition for CCITT Applications, 1991.

[8] CCITT. Draft Recommendation X.711 - Common Manaagement Information Protocol Specification for CCITT Applications, 1991.

[9] IRP e GPP. *EPOS Primer*. Universidade de Stuttgart e GPP - Gesellschaft für Prozessrechner Programmierung, Alemanha, 1983.

[10] ISO. Information Processing Systems - Open Systems Interconnection - Basic Reference Model - ISO 7498, 1984.

[11] ISO. Information Processing Systems - Open Systems Interconnection - Basic Connection Oriented Session Service Definition - ISO 8326, Agosto 1987.

[12] ISO. Information Processing Systems - Open Systems Interconnection - Basic Connection Oriented Session Protocol Specification - ISO 8327, Agosto 1987.

[13] ISO. Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1) - ISO 8824, Dezembro 1987.

[14] ISO. Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1). Extensions to ASN.1. Draft Addendum - ISO 8824/DAD1, 1987.

[15] ISO. Information Processing Systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1) - ISO 8825, Novembro 1987.

[16] ISO. Information Processing Systems - Open Systems Interconnection - Estelle - a Formal Description Technique based on an Extended State Transition Model - ISO/IEC ISO 9074, Novembro 1988.

[17] ISO. CCR Service Definition - ISO 9804 - Final Text, Fevereiro 1990.

[18] ISO. Information Technology - Open Systems Interconnection - Protocol Specification for the Commitment, Concurrency and Recovery Service Element - ISO/IEC 9805, Abril 1990.

[19] ISO. Information Technology - Open Systems Interconnection - Distributed Transaction Processing - Part 1 (Model), Part 2 (Services Definition) and part 3 (Protocol Specification), 1991.

[20] ISO. Information Technology - Open Systems Interconnection - Systems Management Overview ISO/IEC DIS 10040, 1991.

[21] ISO. Information Technology - Open Systems Interconnection - Systems Management (Part 1 - Part 7) - ISO/IEC DIS 10164, 1991.

[22] María Inés Valderrama Restovíc. Compilador ASN.1 e Codificador/Decodificador para BER, 1992.

[23] SUN. *Network Program Manual - Part 3 - Transport Layer Programming*, 1990.

[24] SUN. *System Services Overview - Part 4 - Networking Overview*, 1990.