UTILIZAÇÃO DO AMBIENTE ISODE NO DESENVOLVIMENTO DE APLICAÇÕES DISTRIBUÍDAS

CELSO GONZALEZ HUMMEL
JOSÉ GONÇALVES PEREIRA FILHO
PEDRO FROSI ROSA
STEFANIA STIUBIENER

Dezembro, 1992

ESCOLA POLITÉCNICA DA USP - LSI - PEE - PCS
AV. PROF. LUCIANO GUALBERTO - TRVS 3 , # 158
CEP 05508-900 CX. P. 8174 - SÃO PAULO - SP

Resumo: A camada de aplicação tem uma arquitetura onde os protocolos são vistos como elementos de serviço de aplicação, utilizados para um fim específico. O desenvolvimento de protocolos deste nível é uma área que sempre requereu muito cuidado, pois qualquer aplicação será executada em ambiente distribuído, e, qualquer mau funcionamento, na lógica ou implementação, do protocolo poderá acarretar danos imprevisíveis. O ambiente ISODE [2] aparece como um grande aliado pois oferece um conjunto de protocolos, definidos no modelo de referência OSI, entre eles ACSE, RTSE, ROSE, etc, que podem ser utilizados a partir de uma biblioteca de serviços referentes a cada protocolo [1].

Abstract: *The application layer has an architecture where protocols are treated as application service elements, used with a specific proposal. The protocols development onto this layer is an area which always required careful, remembering any application will run in distributed environment, and, any malfunction in protocol logic and implementation, will bring no predictable result. ISODE environment [2] appears as a great assistant because it provides a set of protocols, defined in the OSI reference model - such as* ACSE, RTSE, ROSE, etc. -, *which can be used as a reference library for each protocol [1].*

1. Introdução

O desenvolvimento de protocolos de comunicação de dados é uma área que sempre requereu muito cuidado, pois qualquer aplicação será executada em ambiente distribuído, e, qualquer mal funcionamento na lógica, ou implementação, do protocolo poderá acarretar danos imprevisíveis.

A camada de aplicação tem uma arquitetura onde os protocolos são vistos como elementos de serviço de aplicação, utilizados para um fim específico. Dentre todos os elementos, o controlador de associações sobressai como um protocolo separado, pois todos os outros protocolos, com raras exceções, deverão se relacionar com ele.

É importante ressaltar que todos os padrões específicos desenvolvidos a partir da década de oitenta (MMS, TP, RTSE, ROSE, etc) prevêem o relacionamento com o controlador citado, o ACSE (*ASSOCIATION CONTROL SERVICE ELEMENT*).

Qualquer aplicação, portanto, será composta de um ou mais protocolos específicos da tarefa ora almejada, mais o protocolo que faz o controle de associações. Isto faz com que todas as implementações lancem mão do ACSE.

Todos os padrões desta camada são do tipo parceiro-a-parceiro, e são definidos a partir de um conjunto de serviços (com as respectivas máquinas de protocolo), sendo que cada aplicação poderá utilizar um perfil adequado às suas necessidades.

Em função dos requisitos apresentados anteriormente, nasceu o projeto ISODE, desenvolvido por Northrop Research and Technology Center, The Wollongong Group Inc, e a University College London. Ele é um projeto que oferece um ambiente de desenvolvimento de aplicações OSI em redes não OSI (Fig. 1.1).

É importante ressaltar que o ISODE não é ainda um produto, mas, um exemplo de projeto e implementação da transição TCP/IP -> OSI. No entanto com a criação do consórcio ISODE em março de 92, com sede nos EUA (Austin) e na Europa (Londres), diversas entidades sem fins lucrativos se reuniram com o propósito de desenvolvimento de produtos. Este consórcio atende a fabricantes, usuários de rede e entidades acadêmicas.

O ambiente ISODE aparece como um grande aliado pois oferece um conjunto de protocolos, definidos no modelo de referência OSI, entre eles o ACSE, que podem ser utilizados a partir de uma biblioteca de serviços referentes a cada protocolo. Além disso, o usuário do ambiente ISODE fica desobrigado de conhecer os protocolos que não sejam pertinentes ao perfil desejado. Por exemplo, o relacionamento dos protocolos específicos com o ACSE é totalmente transparente.

Outro fato que fica transparente ao usuário são as sintaxes abstrata e de transferência: elas são manipuladas pelo ambiente através de um conjunto de estruturas de dados, onde o usuário não precisa se preocupar com os dados de baixo nível.

Além do ISODE apresentar um conjunto de protocolos do modelo de referência OSI, que servem de ponto de partida no desenvolvimento de aplicações distribuídas, existe uma outra vantagem que é a homogeneidade das aplicações, isto é, existe uma

chance maior de interoperabilidade entre as aplicações desenvolvidas a partir do ambiente citado.

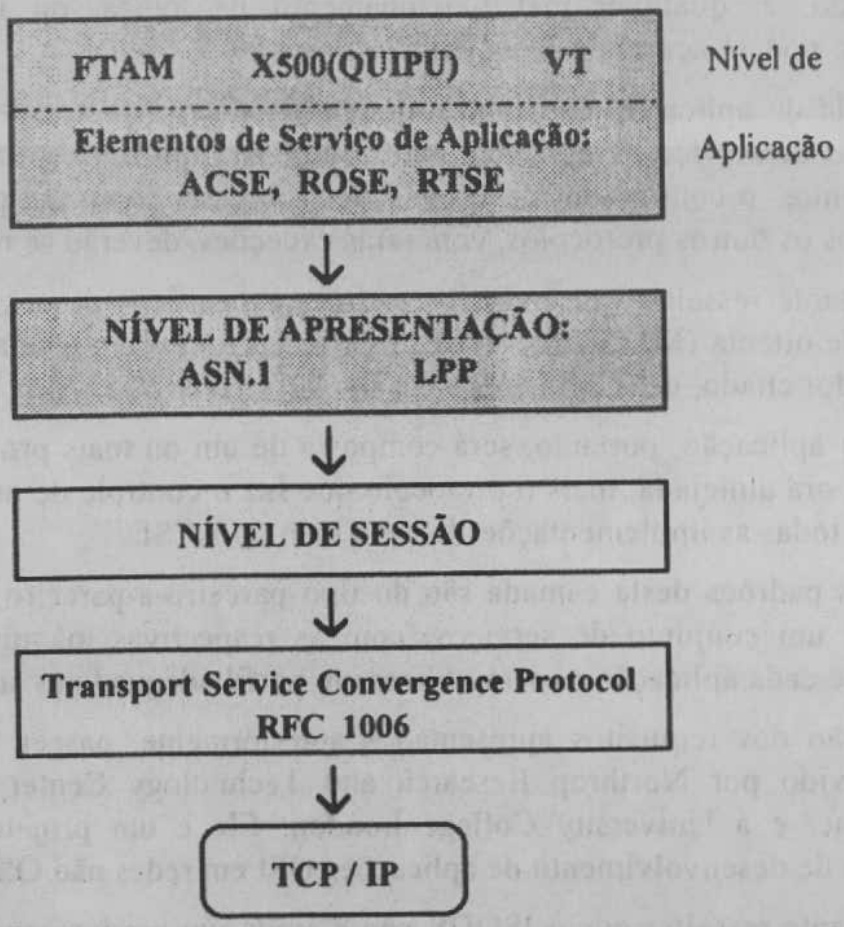

Figura 1.1 - Arquitetura de Protocolos do Projeto ISODE.

Em ambientes distribuídos, além do ACSE, o padrão que define os serviços de diretório de aplicações tem vital importância. Trata-se do protocolo que oferece as facilidades de armazenamento dos atributos das aplicações OSI. Todas as capacidades de um sistema distribuído, que implementa o serviço de diretório, estão contidas em uma base de dados distribuídas, cujas entradas deverão conter um conjunto de atributos, entre eles o nome da aplicação e o seu endereço dentro do ambiente.

A configuração de uma aplicação, dentro do ambiente ISODE, requer a definição da comunidade a qual pertence a referida aplicação. Isto prevê a criação de entradas em tabelas, de modo coerente, para que o relacionamento da aplicação com o ambiente (por exemplo, com o diretório) seja transparente.

Este trabalho tem por objetivo apresentar a utilização do ambiente ISODE no desenvolvimento de uma aplicação envolvendo o protocolo RTSE (transferência confiável), bem como mostrar o relacionamento deste protocolo com o serviço de diretório do ISODE (Quipu). A aplicação é desenvolvida em linguagem "C", sob sistema operacional SUNOS 4.1.1 (compatível com UNIX 4.3.2 BSD) e utiliza a biblioteca ISODE com as primitivas do RTSE.

2. Relacionamento TCP/IP x OSI

Os principais padrões de comunicação aqui considerados são o TCP/IP e o modelo OSI. O modelo OSI e o TCP/IP são padrões de arquiteturas abertas, sendo que, hoje, para muitos especialistas, resiste uma ininputável concorrência. As redes TCP/IP são largamente utilizadas, de uso/interface de programação relativamente simples e com produtos prontamente disponíveis no mercado, isto é, o TCP/IP é um padrão *de facto*, com uma larga base instalada - a rede Internet.

O modelo OSI é uma boa idéia, com produtos muito completos, mas, ainda, com limitada disponibilidade a nível comercial.

2.1. O TCP/IP e as Aplicações OSI

As aplicações voltadas ao TCP/IP têm a vantagem de existirem há muitos anos, por exemplo, muitos usuários estão familiarizados com o FTP, que serve como um mecanismo para transferir arquivos entre nós de uma rede TCP/IP, e com o TELNET, que é utilizado para iniciar uma sessão em sistemas remotos.

As aplicações OSI, que ainda não são largamente utilizadas, são mais completas e mais elaboradas. Por exemplo, o padrão X.400 (MHS - Message Handling System) oferece meios para transmissão de voz e fax, dentre muitas outras funcionalidades, enquanto o protocolo de transferência de E-mail simples (SMTP) do TCP/IP não oferece suporte a tais facilidades. A tabela a seguir mostra as aplicações OSI e TCP/IP.

Função	Aplicação TCP/IP	Aplicação OSI
E-mail	**Simple Mail Transfer Protocol (SMTP)**	**x.400**
Transferência de Arquivo	**File Transfer Protocol (FTP)**	**FTAM**
Terminal Remoto	**Telnet**	**VT**
Gerenciamento de Rede	**Simple Network Management Protocol (SNMP)**	**CMIP e CMIS**

Tabela 2.1 - Aplicações OSI e TCP/IP

2.2. O modelo OSI e a rede INTERNET

A rede INTERNET é o backbone da comunidade TCP/IP. É interessante notar que devido às vantagens recíprocas do modelo OSI e da rede Internet (respectivamente, grande gama de funcionalidades e uma vasta base instalada) o projeto ISODE vem apresentar a solução de "Aplicações OSI sobre TCP/IP"(rede não-OSI). Geralmente, a funcionalidade adicionada à INTERNET é uma precursora da aceitação por muitos usuários TCP/IP; será interessante se isto for verdade também para as funcionalidades OSI adicionadas a esta rede (a rede INTERNET).

3. A Implementação RTSE no ISODE

O RTSE (Reliable Transfer Service Element) é o elemento de serviço da camada de aplicação que tem por objetivo transferir grandes blocos de dados de forma confiável. O RTSE tem a capacidade de lidar com tipos ASN.1 em vez de *strings* de octetos e fornece uma abstração dos complexos serviços da camada de sessão.

No ISODE, a biblioteca *librtsap(3n)* implementa o serviço de transferência confiável. As rotinas residentes nesta biblioteca manipulam uma série de estruturas de dados necessárias para o controle do estabelecimento e liberação de associações. Uma dessas estruturas é a *AEInfo*, que mantém informações sobre as entidades de aplicação participantes de uma associação. No contexto dessas rotinas, o endereço de uma AE no sistema consiste de duas partes: um endereço de apresentação seguido de uma estrutura do tipo *AEInfo*.

```
typedef struct AEInfo {
        PE    aei_ap_title;
        PE    aei_ae_qualifier;
        int   aei_ap_id;
        int   aei_ae_id;
        int   aei_flags;
#define
 ...
} AEInfo, *AEI;
```

aei_ap_title: título do processo da aplicação;

aei_ae_qualifier: qualificador da entidade de aplicação (ex: filestore);

aei_ap_id: identificador de invocação (instância) do processo de aplicação;

aei_ae_id: identificador de invocação (instância) da entidade de aplicação;

aei_flags: indicam quais identificadores de invocação estão presentes.

Os serviços do RTSE são fornecidos para uso em três fases distintas: estabelecimento da associação, transferência dos dados e liberação da associação de aplicação.

O estabelecimento de associação é feito através da primitiva *RtOpenRequest* e utiliza os serviços do ACSE. Uma vez estabelecida, pode ser iniciada a transferência de dados, feita com o uso da primitiva *RtTransferRequest*.

No contexto do RTSE existe a figura da *vêz*. Ao usuário com a vêz é permitido enviar dados ou liberar uma associação, por exemplo.

Para a fase de liberação da associação três primitivas são fornecidas: *RtCloseRequest*, que permite uma liberação negociada, bilateral, *RtUAbort*, que permite liberação unilateral e é usada quando o usuário detecta um erro irrecuperável e *RtPAbort*, usada pelo provedor (*provider*) para liberar de forma abrupta a associação.

4) O serviço de diretório OSI - A implementação QUIPU

O QUIPU é a implementação no ambiente ISODE do serviço de diretório OSI, com base na interpretação da recomendação X.500 do CCITT (norma ISO 9594). A implementação QUIPU provê mecanismos para a manipulação de informações distribuídas por parte de usuários humanos ou aplicações OSI.

Numa interação típica QUIPU/usuário podemos identificar alguns dos agentes do diretório e suas relações com o sistema de arquivo UNIX.

a) DUA (directory user agent): elemento através do qual o usuário se conecta quando interage com o serviço de diretório. É através deste elemento que o usuário acessa informações armazenadas na base de dados mantida pelo sistema. O DUA é configurado no sistema por intermédio do arquivo *dsaptailor*. É neste arquivo que são definidos os endereços de rede dos DSAs com os quais o DUA interage, qual DSA deve ser chamado inicialmente, nome base dos arquivos que contêm as classes de objeto e atributos reconhecíveis, etc. Um exemplo típico é mostrado abaixo:

```
# esta deve ser a primeira linha
oidtable              oidtable

# qual dsa será chamado inicialmente
dsa_address       ofelia      #1024/Internet=143.107.3.208

# caracteristicas de log da instalação
dsaplog               level=exceptions          dflgas=tty
stats                 file=lsi.dua.log

# identificação do inicio do DUA na DIT
local_DIT         c=BR

# formato do identificador de objetos
oidformat         short

# arquivo de iniciação do quipu
quipurc               off

# tamanho da lista/procura do DUA
sizelimit         20
```

b) DSA (directory system agent): entidade que responde às solicitações do DUA, em benefício do pedido do usuário. Este pedido pode ser processado local ou remotamente por um DSA. A interação entre DUA e DSA é governada pelo protocolo DSP (directory service protocol). O estabelecimento de associação é feito através do ACSE e permite a definição de mecanismo de autenticação para o acesso ao diretório. O arquivo *quiputailor* é utilizado como arquivo de configuração para a iniciação de um DSA. A seguir é mostrado um exemplo típico.

```
# nome base dos arquivos OID
oidtable          oidtable
```

```
# nome do DSA
mydsaname  "cn=LSI EPUSP"

# diretório raiz da DIT
treedir          /projetos/isode/isode-7/isode-7-bin/etc/quipu-db
```

c) DIT (directory infomation tree): mantém informações sobre objetos e atributos, definidos com base no conceito de herança de classes. Cada nível da árvore é mapeado em um arquivo UNIX. O nome deste arquivo é determinado unicamente combinando identificadores de nível (*pathname* corrente) com a palavra EDB (entry data block). Como citado acima, a raíz da DIT é definida no arquivo *quiputailor*, no campo *treedir*. O primeiro nível contém entradas que referenciam países, cujo identificador de nível é *c=* (por exemplo, c=BR). O segundo nível referencia organizações de um país, e o identificador é *o=* (por exemplo, o=USP). O terceiro nível armazena informações sobre unidades dentro de organizações, e é identificado por *ou=* (por exemplo, ou=LSI). A figura 5.1, a seguir, ilustra uma DIT.

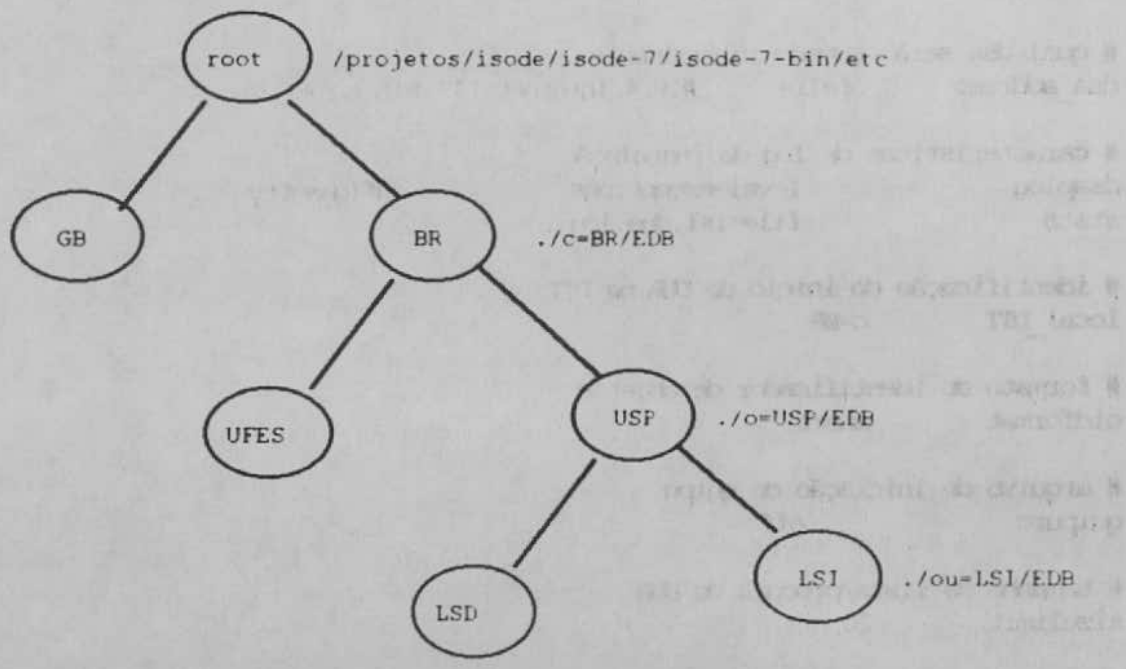

Figura 5.1 - Ilustração de uma Árvore de Informações

d) EDB (entry data block): o conjunto de arquivos EDBs é a implementação lógica definida para a DIT. Cada arquivo EDB contém um cabeçalho, seguido de uma sequência de entradas, como mostrado abaixo.

```
MASTER
9105291854302
o=Universidade de São Paulo
objectClass= top&organization&quipuNonLeafObject
l=Sao Paulo
st=Sao Paulo
streetAddress=Av Prof Luciano Gualberto, trav 3, no 158
...
```

O cabeçalho consiste de duas linhas, onde a primeira contém ou a palavra MASTER, indicando arquivo mestre, ou SLAVE, para arquivo imagem, ou CACHE, para entradas temporárias. A segunda linha contém a versão UTC do arquivo.

Cada entrada em um arquivo EDB contém uma série de atributos. Um atributo, por sua vez, consiste de um tipo de um ou mais valores. O primeiro atributo de uma entrada, que a identifica unicamente, é denominado RDN (relative distinguishing name), por exempo, cn=José da Silva. Outro atributo importante é o que define as classes de objetos às quais a entrada pode pertencer, por exemplo, objectclass=top&organization&quipuNonLeafObject.

Este conjunto de classes de objetos e atributos reconhecíveis no ambiente ISODE é definido nos arquivos *oidtable.oc* e *oidtable.at*, respectivamente, onde *oidtable* é declarado nos arquivos *dsaptailor* e *quiputailor*.

5) Definindo uma nova aplicação no ISODE

a) Descrição da Aplicação

A aplicação consiste de dois programas, instalados em estações distintas, que trocam mensagens entre si visando a transferência de arquivos através do RTSE.

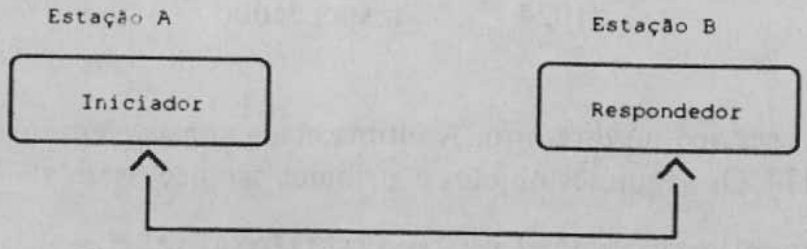

Figura 5.2 - Modelo de Interação entre os programas

b) Definição do Serviço

A catalogação de um novo serviço no ISODE consiste basicamente de quatro etapas envolvendo a manipulação dos bancos de dados *isoentities*, *isobjects* e *isoservices*, além do diretório QUIPU.

Etapa 1:

Definição de uma sintaxe abstrata para o serviço e de um nome para o contexto da aplicação. É utilizado o banco de dados *isobjects*. As entradas neste arquivo apresentam o formato

<descritor do objeto> <identificador do objeto>

mapeando um nome único (distinguishing name) em um identificador de objetos válido no mundo OSI/ISO. O ISODE utiliza a sub-árvore de elementos da ISO:

1.17.2.n.1 *pci* para programa local

1.17.2.n.2 contexto da aplicação para programa local.

Assim, selecionando n=1 para a aplicação descrita, temos:

"isode transfarq1 pci" 1.17.2.1.1
"isode transfarq1 ctx" 1.17.2.1.2

Etapa 2:

Inscrição do serviço no ambiente ISODE. É utilizado o banco de dados *isoentities*. Este arquivo relaciona:

<estação> <serviço> <título da entidade> <seletor tsap>

O ISODE utiliza o identificador 1.17.4.1 para títulos de serviços locais e reserva a faixa de #1024 a #2047 para os seletores tsaps. Selecionou-se, então, o seletor #1024 para a aplicação descrita.

default transfarq 1.17.4.1.1 #1024

Etapa 3:

Relacionamento entre o serviço, o provedor e o programa. É definido no banco de dados *isoservices*, que apresenta o seguinte formato:

<provedor>/<entidade> <seletor> <programa> <args>

Para esta aplicação definiu-se:

"tsap/transfarq" #1024 respondedor

Etapa 4:

Catalogação do serviço no diretório. A última etapa consiste em inscrever o serviço no QUIPU (DIT). Os seguintes objetos e atributos são necessários:

```
objectClass=top&applicationEntitie&quipuObject&iSODEApplication
cn=transfarq
presentationAddress=#1024/Internet=lsi.usp.br
supportedApplicantionContext=isode transfarq1 ctx
acl=
execVector=respondedor
```

5.1. Perfil resulante da Camada de Aplicação

Como se pode observar, uma aplicação distribuída utilizará o serviço escolhido, neste caso o RTSE, o serviço de diretório (QUIPU) e o controlador de associação de aplicação.

Neste caso, a entidade de aplicação consistirá de dois SAOs, um para o QUIPU e o outro para o RTSE, e uma função para disciplinar o relacionamento entre os SAOs exixtentes.

A figura a seguir representa o perfil da camada de aplicação, onde a função de controle dos SAOs é representada, para simplificação, de modo abstrato.

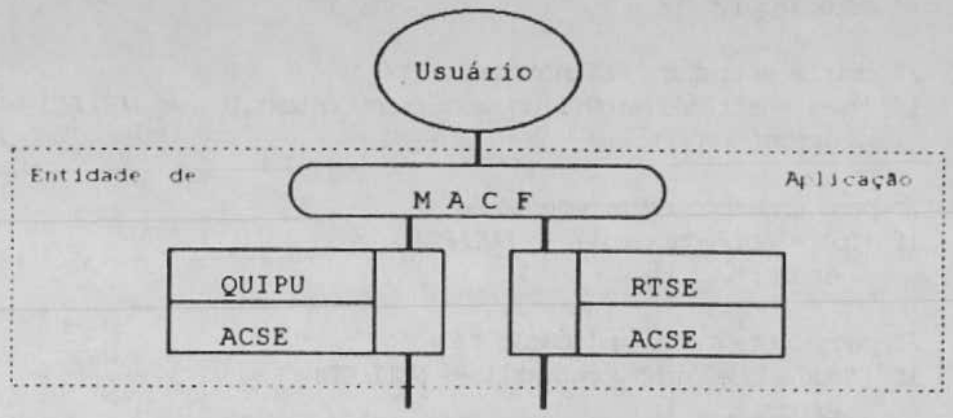

Figura 6.3 - Camada de Aplicação resultante

6. Uma aplicação típica

Neste ítem mostrar-se-á a implementação sob dois ângulos: o do iniciador e o do respondedor. Em ambos os casos apenas as linhas de códigos relevantes serão apresentadas.

Iniciador:

```
#include <stdio.h>
#include "/projetos/isode/isode-7/isode-7-bin/include/isode/rtsap.h"

static char *myservice = "transfarq";                    # nome do serviço
static char *mycontext = "isode transfarq ctx";          # nome do contexto
static char *mypci = "isode transfarq1 pci";             # nome do pci

main(argc,argv,envp)
...

        PE dados;
        int sd,len;
        char nomearq[30],
             hostname[21];
        FILE *p1;
        char *buffer;
        struct RtSAPstart rtss;
        register struct RtSAPstart *rts = &rtss;
        struct AcSAPstart *acs = &rts->rts_start;
        struct PSAPstart *ps = &acs->acs_start;
        register struct PSAPaddr *pa;
        struct PSAPctxlist pcs;
        register struct PSAPctxlist *pc = &pcs;
        register AEI aei;
        register OID ctx,pci;
        struct SSAPref sfs;
        register struct SSAPref *sf;
        struct RtSAPconnect rtcs;
        register struct RtSAPconnect *rtc = &rtcs;
        struct RtSAPindication rtis;
        register struct RtSAPindication *rti = &rtis;
        register struct RtSAPabort *rta = &rti->rti_abort;
        struct AcSAPrelease acrs;
        register struct AcSAPrelease *acr = &acrs;
```

```
/* abre arquivo */

/* inicia estrutura AEIinformation */
if ((aei = str2aei(argv[1],myservice,mycontext,0)) == NULLAEI)
     error( ... );

/* pega endereço de apresentação */
if ((pa = aei2addr(aei)) == NULLPA)
     error( ... );

/* pega contexto de aplicação */
if ((ctx = ode2oid(mycontext)) == NULLOID)
     error( ... );

/* acrescenta uma cópia de ctx à lista */
if ((ctx = oid_cpy(ctx)) == NULLOID)
     error( ... );

/* determina o contexto de apresentação */
if ((pci = ode2oid(mypci)) == NULLOID)
     error( ... );

/* inicia tabela de contexto de apresentação */
pc->pc_nctx = 1;                    /* somente um contexto de apresentação */
pc->pc_ctx[0].pc_id = 1; /* contexto no. 1 */
pc->pc_ctx[0].pc_asn = pci;     /* sintaxe abstrata */
pc->pc_ctx[0].pc_atn = NULLOID;

/* determina UTtime corrente */
if ((sf = addr2ref(gethostname,20))) == NULL) {
     sf = &sfs;
     bzero((char *)sf,sizeof *sf);
}

/* cria PE com nome de arquivo */
dados = oct2prim(nomearq,strlen(nomearq));


/* estabelece associação */
if (RtOpenRequest( ... ))
     error( ... );

if (rtc->rtc_result != RTS_ACCEPT)
     error( ... );

/* pega descritor da associação */
sd = rtc->rtc_sd;
RTCFREE(rtc);
/* transferência do arquivo */
do {
     if ((len = fread(buffer,128,1,p1)) == -1)
          error( ... );

     /* cria primitiva */
     dados = oct2prim(buffer,len);

     /* envia os dados */
     if (RtTransferRequest(sd,dados,NOTOK,rti) == NOTOK)
          error( ... );
```

```
            RTAFREE(rta);
            pe_free(dados);
        } while(!feof(pl));
        fclose(pl);

        /* libera associação */
        if (RtCloseRequest( ... ) == NOTOK)
            error( ... );
        else if (!acr->acr_affirmative) {
            RtUAbortRequest( ... );
            error( ... );
        }
        ACRFREE(acr);
        exit(0);
}
```

Respondedor:

```
#include <stdio.h>
#include "/projetos/isode/isode-7/isode-7-bin/include/isode/rtsap.h"

main(argc,argv,envp)
...
{
        PE dados;
        int sd,len;
        char *nomearq;
        FILE *pl;
        char *buffer;
        struct RtSAPstart rtss;
        register struct RtSAPstart *rts = &rtss;
        struct RtSAPindication rtis;
        register struct RtSAPindication *rti = &rtis;
        register struct RtSAPabort *rta = &rti->rti_abort;
        struct AcSAPstart *acs = &rts->rts_start;
        struct PSAPstart *ps = &acs->acs_start;

        /* recupera estado RTSE */
        if (RtInit(argc,argv,rts,rti) == NOTOK)
            error ( ... );
        sd = rts->rts_sd;
        RTSFREE(rts);

        /* pega o nome do arquivo a ser transferido */
        nomarq = prim2str(rts->rts_data,&len);
        nomearq[len] = 0;

        /* abre arquivo para escrita */

        /* libera estrutura de dados */
        pe_free(rts->rts_data);

        /* responde ao Open Request */
        if (RtOpenResponse( ... ) == NOTOK)
            error ( ... );
```

```
        /* fica aguardando um evento de Transfer ou Connection Close */
        for(;;)
            switch(result = RtWaitRequest(sd,NOTOK,rti)) {
                case NOTOK:
                        error( ... );
                        break;
                case DONE:
                        if (rti->rti_type == RTI_FINISH)
                                if (RtCloseResponse( ... ) == NOTOK) {
                                        error( ... );
                                        fclose(pl);
                                }
                                else
                                        error( ... );
                        break;
                case OK:
                        buffer = prim2str(rti->rti_transfer.rt_data,&len);
                        fwrite(buffer,len,1,pl);
                        pe_free(rti->rti_transfer.rt_data);
                        break;
                default:
                        break;
            }
        exit(0);
}
```

7. Comentários finais

Como se pode notar, o projeto ISODE oferece funcionalidades ao projetista de protocolos que tornam esta tarefa muito menos árdua. O tratamento dos protocolos auxiliares, tais como ACSE, é transparente. O mesmo pode-se dizer das sintaxes abstrata e de transferência (através de funções tais como *str2prim* e *prim2str*).

A maior dificuldade que se impõe inicialmente é o preenchimento das tabelas que cadastrarão os serviço no diretório de aplicações.

O ambiente ISODE oferece bibliotecas, com o conjunto de primitivas necessárias à implementação do protocolo escolhido, que tornam o desenvolvimento dos programas distribuídos mais amistosos ao usuário-programador.

BIBLIOGRAFIA

1. Rose, M. T. The Open Book: a practical perspective on OSI. Ed. Prentice-Hall, 1990

2. Rose, M. T.; et al. ISODE: The ISO development User's Manual. v.1-5 Versão 7.0 - 1991

3. ISO/IEC DIS 8649 Information Processing Systems - Open Systems Interconnection - Service Definition for the ACSE - 1988

4. ISO/IEC DIS 8650 Information Processing Systems - Open Systems Interconnection - Protocol Specification for the ACSE - 1988

5. Rosa, P.F. Utilização da Engenharia de Protocolos na implementação de um protocolo de aplicação em ambienete de manufatura - Dissertação de Mestrado Escola Politécnica da USP - 1992

6. Rose, M. T.; et al. OSI Transport Services on top of the TCP Computer Networks and ISDN, v.12, n.3, 1986

7. Unix Standards. Computer Technology Research Corp. First Edition, Second Printing, may 1992.3.1