ESTRUTURAÇÃO E IMPLEMENTAÇÃO DO PROTOCOLO FTAM

Clizenit, P. A. L.
Joberto, S. B. M.

Grupo de Redes de Computadores
UFPB/DSC/DEE
CP 10117

58.100 - Campina Grande - PB

RESUMO

O objetivo principal deste artigo é apresentar um modelo de implementação dos serviços de transferência, acesso e gerenciamento de arquivo, derivados da especificação do protocolo FTAM proposta pela ISO.

Inicialmente, apresentamos uma visão geral dos serviços FTAM, definindo um conjunto funcional de serviços e uma classe de protocolo a ser implementada. Em seguida, descrevemos um modelo funcional que caracteriza todos os processos envolvidos na implementação e tecemos considerações sobre as estruturas de dados utilizadas.

1. INTRODUÇÃO

Com a evolução tecnológica tem-se obtido uma maior complexidade e funcionalidade nos circuitos integrados, memórias, processadores, equipamentos periféricos, etc. Percebe-se também uma evolução das Comunicações (meios de transmissão, confiabilidade, melhor vazão, etc) e das técnicas de Engenharia de Software (linguagens, métodos de especificação formal, etc).

De uma maneira geral, esta evolução leva a uma maior demanda de compartilhamento da informação pelos usuários nos sistemas de comunicação (redes, sistemas distribuídos ...). Logo, aplicativos do tipo transferência de mensagens, compartilhamento de arquivos, distribuição de tarefas são ferramentas importantes para os usuários.

Um problema relevante a considerar é a heterogeneidade dos equipamentos e, neste sentido, acredita-se que somente uma padronização poderá garantir uma estabilidade para os usuários.

O Modelo de Referência para Interconexão de Sistemas Abertos da ISO ("International Standards Organization") RM-OSI/ISO [1] é uma tendência internacional, endossada a nível nacional pela

Política Nacional de Informática da SEI. Ele define uma padronização para o serviço/protocolo de transferência de arquivo conhecida como FTAM ("File Transfer Access and Management") [2-5].

Neste artigo, apresenta-se uma implementação do protocolo padrão FTAM e discute-se as características principais da implementação.

2. APRESENTAÇÃO GERAL DO SERVIÇO FTAM

O objetivo principal dos serviços de Transferência, Acesso e Gerenciamento de Arquivo (FTAM) é permitir que sistemas heterogêneos realizem operações em arquivos de forma eficaz [6].

O serviço FTAM faz uso de um "arquivo virtual" cujo formato é independente da máquina hospedeira. Na transferência de um arquivo entre dois sistemas, o sistema fonte primeiro mapeia os dados arquivados para o formato do arquivo virtual e em seguida o sistema destino mapeia o arquivo virtual recebido para o seu formato local.

A transformação de arquivos não é da alçada do serviço FTAM, mas sim dos seus sistemas usuários que devem implementar funções de mapeamento que absorvam as diferenças de representação e especificação de arquivos entre o sistema de arquivo virtual e o real. A figura 1 mostra o mapeamento entre o sistema de arquivo real e o ambiente OSI.

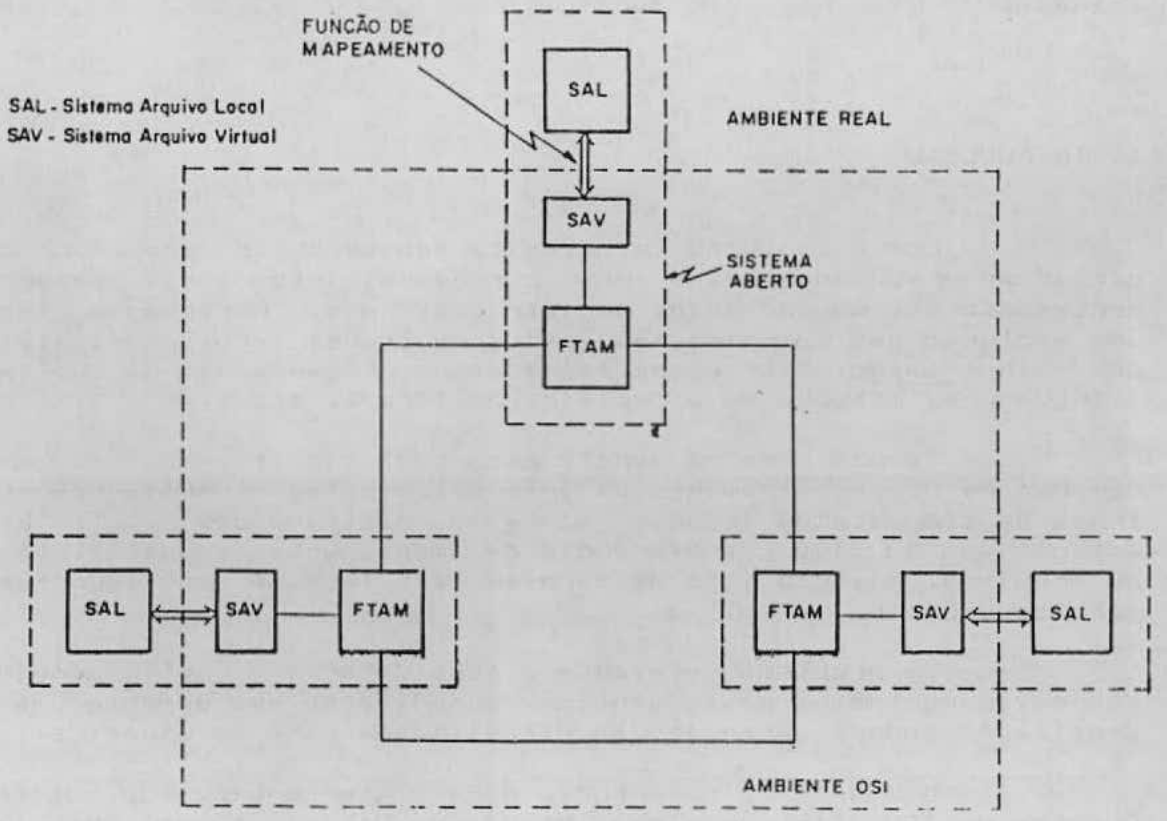


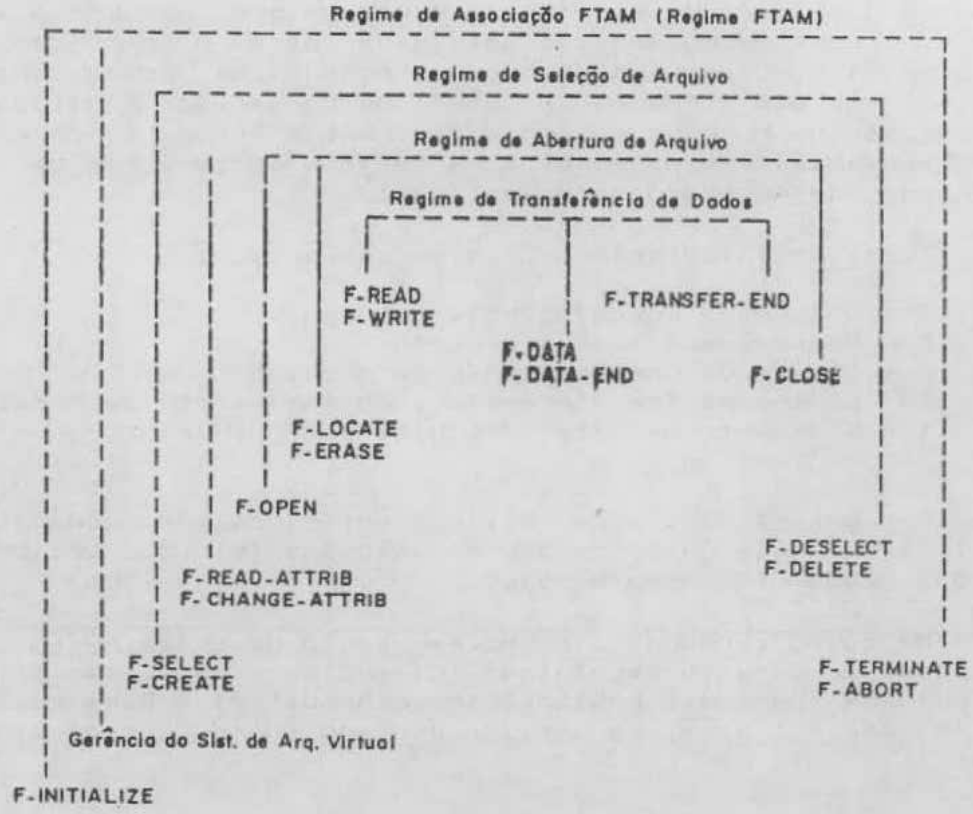
Fig. 1 - Arquitetura de Sistemas Abertos com Serviços de Transferências, Acesso e Gerenciamento de Arquivos

O formato do arquivo virtual é definido pelo serviço FTAM através de um modelo comum de arquivos chamado Sistema de Arquivos Virtual (SAV).

A utilização deste modelo padrão (SAV) para descrever arquivos permite a interconexão de sistemas de diferentes níveis de complexidade e que possuem diferentes maneiras de armazenar e acessar os dados de um arquivo qualquer [7]. O SAV define tanto a estrutura e os atributos associados aos arquivos, como também as operações válidas nos mesmos.

Na descrição dos serviços FTAM, a entidade que inicia e controla a comunicação é chamada entidade iniciadora e a entidade de aplicação parceira, à qual está associado o sistema de arquivo virtual remoto é chamada entidade respondedora.

O serviço de arquivo e o protocolo suportando tais serviços criam, numa série de etapas, um ambiente de trabalho no qual as atividades solicitadas pelo iniciador podem ser realizadas. O período de tempo no qual uma parte do estado comum mantido pelos usuários do serviço permanece válido é chamado regime . Como mais estados compartilhados são progressivamente estabelecidos, um aninhamento de regimes é mantido. A figura 2 mostra todos os regimes definidos e as primitivas de serviço envolvidas.



-Fig.2 - Regimes do Serviço de Arquivo e Primitivos de Serviço envolvidos

O serviço de arquivo é muito rico em funções e uma ampla faixa de tipos de aplicações pode ser suportada. Se fosse permitida uma escolha arbitrária das funções a serem realizadas, haveria no outro lado, uma pequena chance de uma escolha similar ser feita e, consequentemente, uma pequena probabilidade de comunicação.

Para solucionar esse problema, o serviço FTAM define dois tipos de seleção funcional. No mais básico, as funções definidas são agrupadas em UNIDADES FUNCIONAIS. Uma implementação deve suportar uma unidade funcional completamente ou não suportá-la e isto reduz o número de escolhas a serem feitas. São definidos mecanismos que permitem negociação de unidades funcionais quando o regime FTAM é inicializado, e isto faz com que duas entidades comunicantes cheguem a um entendimento comum do conjunto de unidades funcionais disponíveis.

Isto em si reduz a variedade, contudo, ainda deixa considerável liberdade na escolha a ser feita. Uma convergência subsequente é alcançada pela definição de CLASSES DE SERVIÇO . Uma classe de serviço consiste na combinação de um grupo de unidades funcionais para um determinado propósito.

Deste modo em cada inicialização de uma associação de aplicação é necessário negociar a Classe de Serviço, as Unidades Funcionais opcionais e também o nível de serviço, que pode ser Confiável ou Corrigível pelo Usuário.

No serviço de arquivo confiável o usuário especifica seu padrão de qualidade de serviço, mas não se preocupa com a recuperação de erros, delegando tal atividade para o provedor do serviço de arquivo. O serviço de arquivo corrigível pelo usuário inclui primitivas que fornecem aos usuários do serviço facilidades para recuperação de erros e gerência da transferência, tendo estes então, uma flexibilidade na escolha de estilos de gerência de erros, dependendo da necessidade da aplicação.

Foram definidas cinco Classes de Serviço:

T = Classe de Transferência de Arquivo
A = Classe de Acesso a Arquivo
G = Classe de Gerenciamento de Arquivo
TG= Classe de Transferência e Gerenciamento de Arquivo
I = Classe Irrestrita (definida pelo usuário)

A tabela 1 mostra os serviços definidos para cada unidade funcional e quais unidades são mandatórias (m) ou opcionais (o) dentro de cada Classe de Serviço.

No que diz respeito à implementação descrita neste artigo, a classe de serviço escolhida foi a Classe de Acesso a Arquivo, a unidade funcional opcional selecionada foi a Gerência de Arquivo Limitada e o nível de serviço adotado foi o Corrigível pelo Usuário.

UNIDADES FUNCIONAIS	SERVIÇOS	CLASSES DE SERVIÇO T	A	G	TG	I
NÚCLEO	estabelecimento do regime FTAM término ordenado ou abrupto do regime FTAM seleção e desseleção de arquivo	m	m	m	m	m
LEITURA	leitura de massa de dados transf.de unid. de dados fim de transf. dos dados fechamento e abertura de arquivo	*	m		*	o
ESCRITA	escrita de massa de dados transf.de unid. de dados fim de transf. dos dados cancelamento de transferência de dados fechamento e abertura de arquivo	*	m		*	o
ACESSO A ARQUIVO	localizar e apagar FADU	m				o
GERÊNCIA DE ARQUIVO LIMITADA	criar arquivo deletar arquivo ler atributo	o	o	m	o	o
GERÊNCIA DE ARQUIVO AMPLIADA	alterar atributos (requer a anterior)	o	o	m	o	o
AGRUPAMENTO	início/fim de agrupamento	m	o	m	m	o
RECUPERAÇÃO	recuperação de regime ponto de verificação cancel.de transf. de dados	o	o	o	o	o
REINÍCIO DE TRANSF. DE DADOS	recomeço de transferência de dados ponto de verificação cancelamento transf. dados	o	o	o	o	o

TABELA 1: Unidades Funcionais e Classes de Serviço

3. MODELO FUNCIONAL DA IMPLEMENTAÇÃO

O modelo funcional, apresentado em seguida, para a implementação do protocolo e das primitivas de acesso aos serviços do FTAM descreve:

- a funcionalidade dos processos envolvidos;
- a comunicação interprocessos;
- a interface entre camadas adjacentes e
- as estruturas de dados utilizadas.

3.1 Processos

Em relação à estrutura de processos utilizados, tem-se uma estrutura multi-processo com facilidades de gerenciamento dos mesmos (p. Ex.: criação, morte, alocação, ...).

A funcionalidade dos processos é a seguinte (Figura 3):

PGIC - Processo Gerente da Interface e da Camada:

é o processo que gerencia toda a camada, ou seja, os processos PGC e as interfaces com a camada inferior e com a camada superior.

PGC - Processo Gerente de Conexão;

é o processo que implementa a máquina do protocolo FTAM, gerenciando portanto as conexões ativas do protocolo citado.

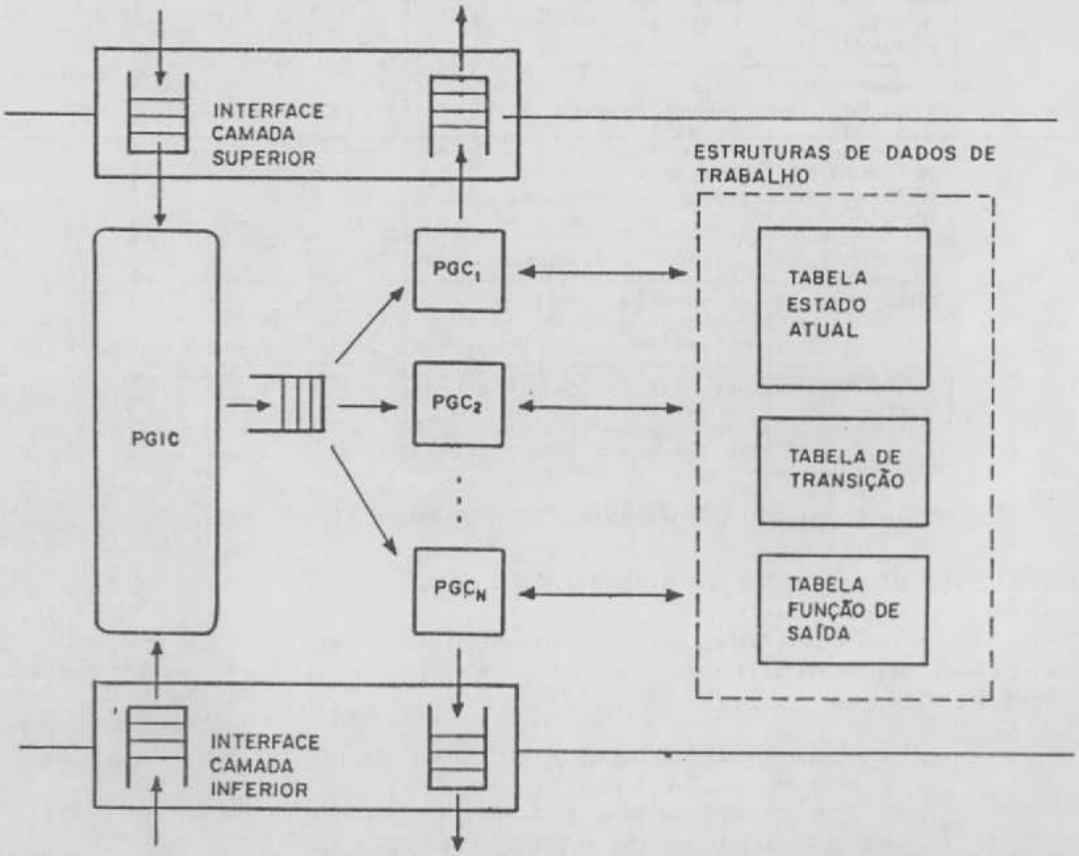


Fig. 3 - Esquema Funcional da Implementação

O processo PGC executa a máquina de protocolo FTAM fazendo a análise de predicados, executando transições de estado e os procedimentos para cada estado da conexão; estimulado por eventos internos e externos à camada.

Existe instância do processo PGC para cada conexão FTAM ativa. O processo PGC usa uma estrutura de dados flexível, compos-

ta pelas tabelas de estado, transição e função de saída discutidas adiante.

Com base nestas estruturas de dados, o processo PGC executa funções, tais como:

- desmontagem e montagem das primitivas de serviço;
- verificação da coerência dos parâmetros das primitivas e das unidades de dados do protocolo (UDPs);
- manutenção de todas as informações referentes ao estado atual da conexão.

O processo PGIC é o responsável pela gerência da camada, incluindo assim, a gerência dos demais processos envolvidos e a gerência da interface entre camadas adjacentes. Em resumo, o PGIC executa as seguintes tarefas básicas:

- criação dinâmica de processos PGC na medida da necessidade da camada, ou seja, em função do número de usuários ativos;
- criação das filas utilizadas para comunicação com as camadas inferior e superior;
- monitoração dos recursos utilizados do sistema operacional (filas, memória compartilhada, ...) e
- recepção de mensagens na interface e roteamento para o processo PGC gerente da conexão correspondente.

O PGIC se sincroniza e troca parâmetros com os PGCs via a tabela de estado a qual é posta num segmento de memória compartilhada ("shared memory") também gerenciado pelo PGIC.

A passagem de primitivas de serviço de entrada do PGIC para os PGCs se dá através de um mecanismo de comunicação interprocesso de filas.

A figura 4 apresenta o fluxograma que descreve em resumo as ações executadas pelo processo PGIC para a execução de suas tarefas.

3.2 Interface com Camadas Adjacentes

A interface com as camadas adjacentes nesta implementação é composta por quatro filas de mensagens (Figura 3):

- fila entrada superiror;
- fila entrada inferior;
- fila saída superior e
- fila saída inferior.

3.3 Estruturas de Dados

As estruturas de dados básicas usadas na implementação do FTAM são as seguintes:

- Tabela de Estado Atual (TEA)
- Tabela de Transição (TT)
- Tabela de Função de Saída (TFS)

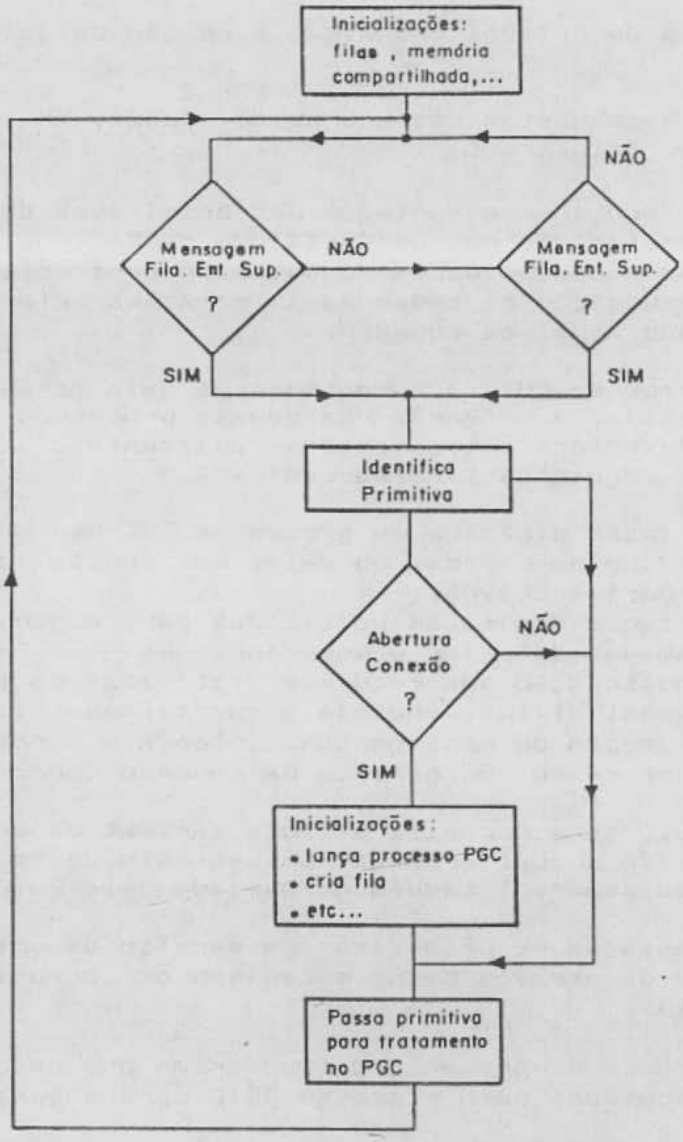


Fig.4 - Fluxograma Geral Simplificado do Processo de Gerência da Camada

A tabela de estado atual contém todos os parâmetros de estado da conexão FTAM, semáforos e parâmetros necessários para a sincronização e comunicação do processo PGIC com o processo PGC tais como:

- identificadores de filas
- buffers de trabalho
- endereços
- classes e nível de serviço
- unidades funcionais
- atributos do arquivo
- diagnósticos
- etc...

O anexo 1 ilustra parte da estrutura de dados da tabela de estado atual em linguagem C.

A tabela de transição associa os eventos ocorridos ao estado atual da conexão. Assim sendo, à partir da tabela de transição são inicialmente ativadas as funções que testam os predicados para o evento ocorrido para, em seguida, serem executadas as funções ou ações de saida. A tabela de transição aponta também para o conjunto de ações de saida em função do teste de predicados executado. Neste caso, um arranjo de funções de saida é selecionado à partir do valor de retorno das funções de teste de predicados.

A figura 5 ilustra a estrutura da tabela de transição utilizada.

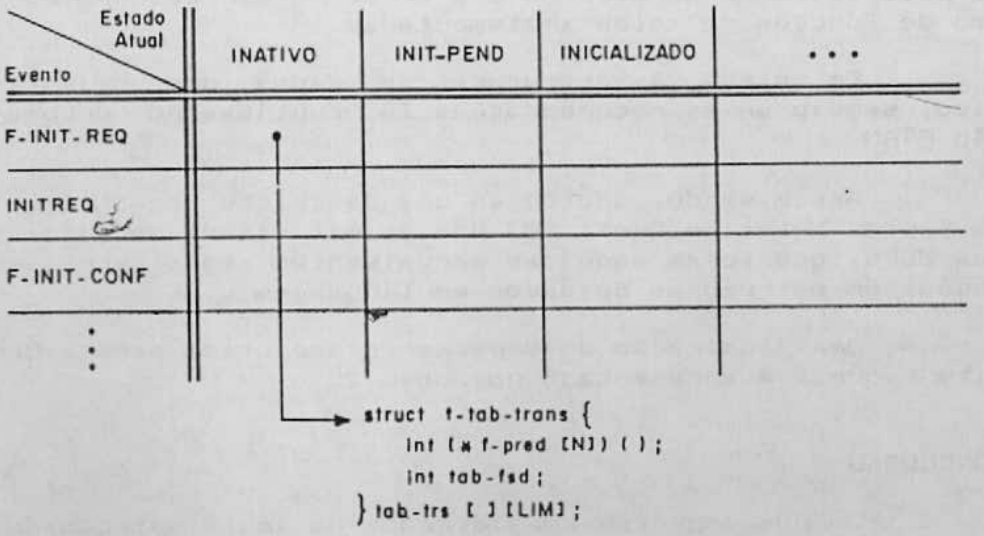


Fig. 5 - Estrutura de Dados da Tabela de Transição

Tem-se um arranjo bidimensional indexado pelos eventos e estados de conexão do FTAM. Para cada dupla evento/estado tem-se um arranjo de apontadores de funções de teste de predicado que são executadas sequencialmente e um inteiro indexando uma entrada da tabela de função de saida descrita em seguida.

A tabela de função de saida lança todas as ações pertinentes à transição ocorrida. Assim sendo, para cada dupla evento/estado, que leva a uma transição, existem arranjos de funções de saida. A seleção de qual arranjo de funções é acionado, levando à execução de um conjunto de ações de número variável, é determinada pelo valor de retorno das funções de teste de predicado acionadas através da tabela de transição.

Uma ilustração de uma entrada da tabela de função de saída é apresentada no exemplo abaixo:

```
int (*f_ac [ ] [9] [7]) ( ) = {

    {{NULL},{inirq_pdu,ac1_rf,ac3_rf,init_pd,NULL},
     {f6_inicf_pr,closed,NULL}},
    .
    .
    .
    .

};
```

Tem-se um arranjo tridimensional de apontadores de função. O primeiro nível é indexado pela dupla evento/estado, o segundo pelo teste de predicados e o terceiro corresponde ao numero máximo de funções de saída implementadas.

Em relação às estruturas de dados das primitivas de serviço, seguiu-se as recomendações ISO contidas no documento do padrão FTAM.

Assim sendo, adotou-se uma descrição em ASN.1 (" Abstract Syntax Notation One") [8] das primitivas de serviço, bem como das UDPs, que foram mapeadas manualmente (sem ferramentas de tradução) em estruturas de dados em linguagem C.

Uma ilustração do mapeamento executado para a UDP initialize request é apresentado no anexo 2.

4. CONCLUSÃO

Um dos aspectos considerados na implementação do protocolo FTAM foi o seguimento das especificações do padrão conforme definido pela ISO. Para possibilitar a realização de tal objetivo, utilizou-se um sistema computacional do porte de um supermicro rodando um sistema operacional com recursos suficientes. Tal base de desenvolvimento permitiu a implementação do padrão FTAM sem as"simplificações" que são algumas vezes feitas para acomodar a ausência de recursos de determinados sistemas operacionais.

No que diz respeito ao modelo funcional, implementou-se uma estrutura flexível para os processos e para os dados.

A estrutura de processo utilizada permite a implementação de funções de gerência mais eficientes, permitindo inclusive a implementação de um protocolo de gerência de camada de rede. Em relação à implementação realizada, destacam-se as seguintes facilidades:

- alocação dinâmica de processos;
- controle dos recursos do sistema operacional do computador e
- pseudo paralelismo na execução o protocolo por conexão, caracteristica esta que espera-se poder repercutir positivamente no desempenho da implementação.

Em relação às desvantagens da estrutura de processos utilizada, destaca-se a limitação do número de conexões ativas em função da própria limitação dos recursos do sistema operacional (p. ex.: número máximo de processos ativos, bufferização de filas, número de filas, ...). é importante observar que, normalmente, estes recursos são parametrizáveis, podendo, portanto, serem ajustados.

As estruturas de dados implementadas permitem a implantação gradual das classes funcionais do FTAM sem modificações na implementação das classes já desenvolvidas e instaladas. Em efeito, a anexação de uma nova classe é feita com a introdução de novas funções através da inicialização de apontadores de função nas tabelas de transição e de função de saída. Na realidade, a estrutura de dados foi dimensionada para suportar a implementação do pacote FTAM completo.

Para concluir, fez-se uma implementação em linguagem C com o sistema operacional " Unix - System V " dispondo de recursos de comunicação interprocesso tais como filas, memória compartilhada e semáforos.

5. REFERÊNCIAS BIBLIOGRAFICAS

[1] ISO IS 7498 - "Information Processing Systems - Basic Reference Model for Open System Interconnection", 1983.

[2] ISO DIS 8571/1 - Open Systems Interconnection - File Transfer, Access and Management - Parte 1: General Introduction - 1987

[3] ISO DIS 8571/2 - Open Systems Interconnection - File Transfer, Access and Management - Parte 2: The Virtual Filestore Definition - 1987

[4] ISO DIS 8571/3 - Open Systems Interconnection - File Transfer, Access and Management - Parte 3: The File Service Definition - 1987

[5] ISO DIS 8571/4 - Open Systems Interconnection - File Transfer, Access and Management - Parte 4: The File Protocol Specification - 1987

[6] MOURA, José Antão Beltrão et. all - Protocolos de Alto Nivel e Avaliação de Desempenho - McGraw-Hill - 1986

[7] CARVALHO, Tereza C.M.B. - Serviços de Transferência de Arquivo em Redes Locais - Dissertação de Mestrado - Escola Politécnica da USP - 1988

[8] ISO 8824 - Information Processing Systems - Open Systems Interconnection - Abstract Syntax Notation 1 - ASN.1

[9] THOMAS, Rebecca et. all - Advanced Programmer's Guide to UNIX - System V - 1986

ANEXO 1 - Tabela de Estado Atual

```
struct t_est_at {
        int         idf_assc;           /* ident. da associação */
        int         pr_gr_cnx;          /* ident. do proc. gerente  da
                                           conexão */
        int         idfila;             /* ident. da fila do  PGC */
        int         f_s_sup;            /* fila de saida superior */
        int         f_s_inf;            /* fila de saida inferior */
        int         est_cnx;            /* estado da conexão */
        int         sinal;              /* sinal de que existe msg  na
                                           fila */
        int         flag;               /* flag de entrada livre(0) ou
                                           ocupada (1) */
        struct    t_msg  *msgp;         /* apontador para o buffer in-
                                           termediário do pgc */
        struct    t_p_ftam;             /* parâmetros do FTAM */
};


struct t_p_ftam {
        char        t_apl_ch;           /* título da aplicação chama-
                                           da */
        char        t_apl_ct;           /* título da  aplicação  cha-
                                           mante */
        int         e_ap_ch;            /* endereço  de  apresentação
                                           chamado */
        int         e_ap_ct;            /* endereço  de  apresentação
                                           chamante */
        BOOLEAN     g_cont_ap;          /* gerência  do  contexto  de
                                           apresentação */
        int         niv_serv;           /* nível de serviço */
        int         clas_serv;          /* classe de serviço */
        BITSTRING   und_func;           /* unidades funcionais */
        BITSTRING   gp_atb;             /* grupo de atributos */
        int         rbck;               /* disponibilidade  de  retro-
                                           cesso */
        int         q_s_com;            /* qualidade  do  serviço  de
                                           comunicação */
        struct      t_lst_cont lst_cont;/* lista tipo de conteúdo */
        G_STRING    id_inic;            /* identidade do iniciador */
        G_STRING    cont;               /* contabilidade */
        union       t_senha senh;       /* senha do sistema de arqui-
                                           vo */
        int         chkp;               /* janela de checkpoint */
        int         r_est;              /* resultado de estado */
        int         r_ac;               /* resultado de ação */
        char        t_apl_r;            /* título da  aplicação  res-
                                           pondente */
        int         e_ap_r;             /* endereço  de  apresentação
                                           respondente */
        struct      t_diagn diagn;      /* diagnóstico */
};
```

ANEXO 2 - UDPs em ASN.1

a) Descrição da UDP "initialize-request" em ASN.1:

```
F_INITIALIZE_request ::= SEQUENCE {

  protocol-id    Protocol-Version
       DEFAULT  {major {version-1}, minor revision-1},
  presentation-context-management [1] IMPLICIT BOOLEAN
       DEFAULT  FALSE,
  service-level    Service-Level  DEFAULT reliable,
  service-class    Service-Class  DEFAULT transfer-class,
  functional-units    Functional-Units  OPTIONAL,
  attribute-groups    Attribute-Groups  DEFAULT { },
  rollback-availability  Rollback-Availability  OPTIONAL,
  contents-type-list  Contents-Type-List  OPTIONAL,
  initiator-identity  User-Identity  OPTIONAL,
  account    Account    OPTIONAL,
  filestore-password  Password    OPTIONAL,
  checkpoint-window [8] IMPLICIT INTEGER DEFAULT 1
}


Protocol-Version ::= [0] IMPLICIT SEQUENCE {

   major [0] IMPLICIT BITSTRING  {version-1 (0)},
   minor [1] IMPLICIT INTEGER    {revision-1 (1)},
             DEFAULT  revision-1

Service-Level  ::= [2] IMPLICIT INTEGER
                       {reliable (0), user-correctable (1)}

Service-Class  ::= [3] IMPLICIT INTEGER
                       {transfer-class (0),
                        access-class (1),
                        management-class (2),
                        transfer-and-management-class (3),
                        unconstrained-class (4)}

Functional-Units ::= [4] IMPLICIT BITSTRING
                          {read (2),
                           write (3),
                           file-access (4),
                           limited-file-management (5),
                           enhanced-file-management (6),
                           grouping (7),
                           recovery (8),
                           restart-data-transfer (9)}

Attribute-Groups ::= [5] IMPLICIT BITSTRING
                         {storage (0),
                          security (1),
                          private (2)}

Rollback-Availability ::= [6] IMPLICIT INTEGER
                              {no-rollback (0),
                               rollback-available (1)}
```

```
Contents-Type-List ::= [7] IMPLICIT SEQUENCE {

   document-types [0] IMPLICIT SEQUENCE OF Document-Type-Name,
   constraint-sets-and-abstract syntaxes [1] IMPLICIT SEQUENCE {
       constraint-sets [0] IMPLICIT SEQUENCE OF Constraint-Set-Na-
me,
       abstract-syntaxes [1] IMPLICIT SEQUENCE OF
                                        Abstract-Syntaxe-Name}
   }
```

b) Descrição da UDP "initialize-request" em linguagem C:

```
#define    TEX     85

typedef    int     BITSTRING;
typedef    char    BOOLEAN;
typedef    char    G_STRING [TEX];
typedef    char    O_STRING;

/*
 * PDU F-INITIALIZE-request
 */

struct t_lst_cont {
       G_STRING  n_t_dc;      /* nome do tipo do documento */
       struct {
          G_STRING n_cj_r;    /* nome do conjunto de restrições */
          G_STRING n_st_a;    /* nome da sintaxe abstrata */
       } cj_stx;
};

union t_versao {
      BITSTRING p_vers;
      int p_rev;
};

union t_senha {
      G_STRING g_senh;
      O_STRING o_senh;
};

struct t_in_req {
       int id_pdu;                  /* identificador da pdu */
       union t_versao versao;       /* versão do protocolo */
       BOOLEAN g_cont_ap;           /* gerência do contexto de  apre-
                                       sentação */
       int niv_serv;                /* nível de serviço */
       int clas_serv;               /* classe de serviço */
       BITSTRING und_func;          /* unidades funcionais */
       BITSTRING gp_atb;            /* grupo de atributos */
       int rbck;                    /* retrocesso */
       struct t_lst_cont lst_cont/* lista do tipo de conteúdo */
       G_STRING id_i;               /* identidade do iniciador */
       G_STRING cont;               /* contabilidade */
       union t_senha senh;          /* senha do sistema de arquivo */
       int chkp;                    /* janela de checkpoint */
};
```