UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE SOFTWARE PARA DISTRIBUIÇÃO DE ARQUIVOS RECEBIDOS POR E-MAIL VIA INTERNET.

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

EDUARDO JEAN REGIS

BLUMENAU, JUNHO/2000

2000/1-14

PROTÓTIPO DE SOFTWARE PARA DISTRIBUIÇÃO DE ARQUIVOS RECEBIDOS POR E-MAIL VIA INTERNET.

EDUARDO JEAN REGIS

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Sérgio Stringari — Orientador

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Sérgio Stringari

_____

Prof. Antônio Carlos Tavares

_____

Prof. Miguel A. Wisintainer

AGRADECIMENTOS

Agradeço a minha mãe, irmãos e amigo que sempre me apoiaram, incentivando-me nos momentos de desânimo e de alegria e dando-me forças para continuar o meu caminho.

Ao professor e orientador Sérgio Stringari, que durante os meus estudos sempre me direcionou à qualidade com brilhantismo fazendo críticas e elogiando cada passo de meus estudos.

Aos professores agradeço pelos conhecimentos que me foram passados, pelas dedicações que foram dadas, pelas orientações que me ajudaram a chegar até aqui e que seguirão comigo tanto na vida pessoal quanto profissional.

A empresa Cia. Hering no qual nesses dez anos sempre incentivou a estudar.

À Deus e todas as formas de vida que orientam o meu caminho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS E ABREVIATURAS

ARPANET- *Advanced Research Projects Agency NETwork*

DoD - *Departament of Defense*

FTP – *File Transfer Protocol*

IAB - *Internet Activity Board*

IEEE – *Institute of Electrical and Electronics Engineering*

IEN - *Internet Engineering Notes*

IP – *Internet Protocol*

ISO – *International Organization for Stantardization*

MILNET – *MILitary NETwork*

OSI – *Open System Interconnection*

POP – *Post Office Protocol*

RFC – *Request for comments*

SMTP – *Simple Mail Transfer Protocol*

TCP – *Transmission Control Protocol*

UDP – *User Datagragram Protocol*

RESUMO

Este trabalho apresenta um estudo sobre os protocolos Post Office Protocol (POP) e File Transfer Protocol (FTP) apresenta também, considerações sobre os Protocolos Simple Mail Transfer Protocol (SMTP) e Socket, utilizados como tecnologias no desenvolvimento de um protótipo de software para distribuição de arquivos recebidos por *e-mail* via Internet.

ABSTRACT

This work presents a study on the protocols Post Office Protocol (POP), File Transfer Protocol (FTP), show too Protocolo Simple Mail Transfer Protocol (SMTP) and Socket, used as technologies in the development of a software prototype for distribution of files received by e-mail through Internet.

1 INTRODUÇÃO

Nos anos 70 e início dos anos 80, foram desenvolvidos sistemas de correio eletrônico denominados sistemas de tratamento de mensagens computadorizados. Sendo o correio eletrônico uma das ferramentas básicas da automação de escritórios, está se tornando um pré-requisito para a produtividade e a redução de custos nas atividades de troca de mensagens. Mais recentemente, os sistemas de correio eletrônico começaram a ser vistos como uma ferramenta de base sobre a qual devem residir aplicações de automação de escritórios.

Em toda parte do mundo é utilizado a Internet para a transmissão de mensagens, devido sua facilidade de implementação e baixo custo.

A Internet possui algumas tecnologias específicas como, por exemplo, *e-mail* (correio-eletrônico) e transferência de arquivos.

O *e-email* da Internet utiliza o protocolo Simple Mail Transfer Protocol (SMTP) para transferência de mensagens recebidas através de correio eletrônico de um equipamento para outro equipamento. O protocolo Post Office Protocol (POP) é usado por usuários de correio eletrônico para obter as mensagens recebidas e arquivadas em servidores de correio eletrônico ([COM1995]).

A transferência de arquivos na Internet utiliza o protocolo File Transfer Protocol (FTP). Segundo [CAR1994], trata-se de um utilitário de uso interativo que pode eventualmente ser chamado por programas para efetuar a transferência de arquivos.

Diversos softwares de correio eletrônico conhecidos no mercado, como o *OUTLOOK* da Microsoft, utilizam o protocolo POP. O POP serve para que os clientes do servidor de *e-mail* possam buscar as mensagens postadas na sua caixa postal.

O software desenvolvido neste trabalho utiliza esse protocolo (POP) para automatizar o processo de busca de mensagens com arquivos anexados *(Attachment)* no servidor de *e-mail*. O software identifica através do cabeçalho das mensagens, o servidor destinatário e transmite o arquivo através do protocolo FTP. Esse software pode ser utilizado em substituição, por exemplo, nas empresas que utilizam Redes Públicas, como a RENPAC (Rede Nacional de Pacotes), para o recebimento de arquivos comerciais, pela rede Internet.

A Internet utiliza o protocolo TCP/IP, no qual existem investimentos na confiabilidade e desempenho, enquanto as Redes Públicas que utilizam o protocolo X.25 considerado um protocolo confiável, porém um protocolo antigo e que foi especificado numa época em que

os meios de comunicação não eram muito confiáveis, ocasionando a perda de tempo na confirmação de cada pacote transmitido.

O software desenvolvido pretende-se implantar na empresa Cia.Hering, que ficará executando num computador com sistema operacional *Windows 95* e poderá ter configurações como:

a) Tempo de verificação de novas mensagens recebidas no servidor de *e-mail*;

b) Padronização do cabeçalho;

c) Endereços dos arquivos recebidos para ser transferido para os *hosts* responsáveis pelas informações.

A cada instante, conforme configuração do protótipo do software proposto, o software analisará se existe uma nova mensagem com arquivo anexado e, quando for o caso, fará a transferência para o servidor destinatário.

Para a especificação do software, foi utilizada a fluxogramação, considerada por [MEN1989] como sendo uma forma de especificação semi-formal, através de simbologia gráfica.

O ambiente para a implementação do software foi o Delphi 3.0 da Borland Inprise.

1.2 OBJETIVOS

O objetivo principal deste trabalho foi utilizar os protocolos POP e FTP na implementação de um software que distribua arquivos recebidos pela Internet e enviá-los por FTP para os Servidores.

1.3 JUSTIFICATIVA

A importância desse trabalho está em apresentar um software para automatizar o serviço de troca de informações entre organizações, ou seja, gerenciar o envio e o recebimento de arquivos, utilizando como meio de comunicação mundialmente conhecido à Internet. A Internet possui algumas características relevantes para implementação desse software, como baixo custo, facilidade de implementação, grandes investimentos para melhoria da Internet e popularização.

Este trabalho, pretende também servir como uma fonte de consulta para os profissionais interessados em conhecer maiores detalhes relacionados aos protocolos File

Transfer Protocol (FTP), Simple Mail Transfer Protocol e Post Office Procol (POP). Um dos objetivos principais do trabalho, é permitir conhecer os elementos componentes destes protocolos, suas funções e características específicas.

Esse software também tem o objetivo de substituir os serviços públicos como RENPAC e TRANSDATA devido seu alto custo por byte transmitido e por essas tecnologias não serem flexíveis.

1.4 ORGANIZAÇÃO DO TRABALHO

O Capítulo 1 apresenta a estrutura geral do trabalho: Introdução, os objetivos e as justificativas dos assuntos abordados.

O capítulo 2 trata especialmente da Arquitetura Internet: conceitos, características, modelos e protocolos de Aplicação.

O capítulo 3 é voltado as tecnologias envolvidas no protótipo.

O capítulo 4 apresenta as técnicas e ferramentas utilizadas.

O capítulo 5 é voltado ao desenvolvimento do protótipo, onde se apresenta a especificação e implementação do mesmo.

No capítulo 6 é dedicado às conclusões e sugestões de continuidade do trabalho.

E por fim, o capítulo 7 apresenta as bibliografias utilizadas para todo o desenvolvimento deste trabalho.

2 INTERNET

A Internet é uma vasta rede de computadores, constituída por várias pequenas redes que, quando conectadas, se estendem por todo planeta [BRO1995].

Dentro da Internet existe uma verdadeira comunidade virtual, formada por todas as pessoas que usam essas redes de computadores com os mais diversos intuitos: troca de mensagens no correio eletrônico, debates ao vivo (IRC – Internet Relay Chat), grupos de discussão (Usenet e Mailing List), entretenimento (jogos), e até comércio. Tamanha é a riqueza de possibilidades que criou-se o conceito de *cyberspace*, ou espaço cibernético, um mundo eletrônico paralelo altamente democrático e solidário, onde já começa a se formar uma base de conhecimento universal [LUN1997].

Os padrões da Internet não são criados por órgãos internacionais de padronização, como a International Organization for Stantardization (ISO) ou o Institute of Electrical and Electronics Engineering (IEEE), mas ela é uma arquitetura muito aceita, sendo chamada por isso como padrão "de facto", ao contrário do modelo Open System Interconnection (OSI), considerado padrão "de jure". O corpo técnico que coordena a elaboração de protocolo e padrões da Internet é o IAB (Internet Activity Board). A documentação dos trabalhos, propostas para novos protocolos ou alteração de outros já existentes é feita através de artigos conhecidos como Request for Comments (RFC). Propostas ainda em estudos são chamadas de IEN (Internet Engineering Notes) ou *Internet Drafts*. Os RFC’s são numeradas seqüencialmente e em ordem cronológica.

2.1 ORIGEM DA INTERNET

A grande rede das redes tem como pai a paranóia da guerra fria. Nasceu em 1969 como ARPANET(Advanced Research Projects Agency NETwork), com o objetivo de se ter uma rede interligando várias universidades e órgãos do governo de maneira descentralizada ([CAR1994]).

A ARPANET foi muito bem sucedida, e em 1983, todas as universidades dos Estados Unidos quiseram participar. Houve a necessidade de racionalização e divisão em MILNET – MILitary NETwork (localidades militares) e a nova e menor ARPANET (localidades não militares).

O plano de usar a ARPANET não funcionou por várias razões, algumas técnicas, outras políticas. Assim, a *National Science Foundation* (NSF), sem desistir de estabelecer um

novo império político, construiu sua própria e muito mais rápida rede para conectar os centros de supercomputadores: a *National Science Foundation NETwork* (NSFNET).

A NSFNET funcionou como uma mágica. Na verdade, por volta de 1990, tantos "negócios" haviam se transferido da ARPANET para NSFNET que, depois de quase 20 anos, a ARPANET perdeu a sua utilidade e foi desativada ([LEV1995]).

No final da década de 80, enquanto a Internet estava crescendo no meio acadêmico, outro tipo de evolução estava acontecendo. As empresas iniciavam a troca de seus *mainframes* por redes locais de computadores, que fizeram mais do que apenas economizar dinheiro. Elas modificaram a maneira das pessoas trabalharem, multiplicando as possibilidades de lidar com a informação.

Hoje as redes estão se interligando através da Internet, utilizando o computador para navegar no mar de informações existente dentro do *cyberspace*, compartilhando sinais inteligentes com o resto do mundo ([LUN1997]).

2.2 A ARQUITETURA INTERNET

Em 1973, a Agência de Projetos de Pesquisa Avançada para Defesa, Defense Advanced Research Projects Agency (DARPA), decidiu pela necessidade de novos protocolos *host-to-host* na ARPANET. Estes novos protocolos deveriam suportar diferentes técnicas de comunicação. As novas implementações foram completadas em 1974. Em 1978, após quatro anos de testes e refinamentos, o Departamento de Defesa norte-americano, Departament of Defense (DoD) promulgou versões de Transmission Control Protocol Internet Protocol (TCP/IP) e obrigou o seu uso como padrão do DoD.

A arquitetura Internet é composta por dois protocolos principais: o Internet Protocol (IP), responsável pelo encaminhamento de pacotes de dados entre diversas subredes desde a origem até o destino, e o Transmission Control Protocol (TCP) que tem por função o transporte fim-a-fim confiável de mensagens de dados entre dois sistemas.

O IP é um protocolo do tipo datagrama, operando, portanto, no modo não-orientado à conexão, enquanto o TCP é um protocolo orientado a conexão. O conjunto TCP/IP pode, desta forma, oferecer um serviço de alta confiabilidade. Para uso em redes de alta qualidade, onde o problema de confiabilidade não assume grande importância, foi definido o protocolo

User Datagragram Protocol (UDP) que opera no modo não-orientado à conexão e possui funcionalidades bem mais simplificadas que o TCP.

Completando a Arquitetura Internet, é especificada uma camada usuária do TCP ou do UDP, referente às aplicações, tais como correio eletrônico, transferência de arquivos , terminal virtual, entre outras. Além disso, é permitido ao usuário o desenvolvimento de suas próprias aplicações através da utilização de primitivas, especificando de acordo com o conceito de interfaces *sockets.*

De acordo com [CAR1994] as interfaces *sockets* definem, portanto, um padrão para a determinação dos pontos finais de comunicação entre dois sistemas, para o estabelecimento das conexões e para a troca de dados entre eles. Constituindo-se de funções que acessam diretamente a camada de transporte, TCP ou UDP, escondem das aplicações toda a complexidade envolvida nos respectivos protocolos.

Na Arquitetura Internet é também implementado o conceito de portas que são endereços associados às aplicações operando em um sistema. Os endereços dessas portas são especificados pela aplicação em seu domínio de abrangência. Para as aplicações globais no domínio Internet como é o caso do correio eletrônico, da transferência de arquivos e de outras, existe um conjunto de endereços de portas reservadas. As portas mais comuns são:

23 - Serviço de TELNET (emulação de terminal)
25 - Serviço de SMTP (Simple Mail Transfer Protocol - correio eletrônico)
49 - Login Host Protocol
69 - TFTP (Trivial File Transfer Protocol)
70 - Serviço de Gopher
79 - Serviço de Finger
80 - HTTP (HypertText Transfer Protocol)
110 - POP3 (Post Office Protocol 3 - correio eletrônico)
137 - NetBIOS NS (Serviço de nomes NetBIOS)
139 - NetBIOS DS (Serviço de datagrama NetBIOS)
156 - SQL Server (Servidor SQL)
179 - BGP (Border Gateway Protocol - protocolo de roteamento 3 Com)
6667 - IRC (Internet Relay Chat)

Conforme [TAN1996], a arquitetura Internet se baseia em um modelo com quatro camadas, de acordo com a figura 2.2, onde cada uma executa um conjunto bem definido de funções de comunicação. No modelo em camadas da Internet, não existe uma estruturação

formal para cada camada, conforme ocorre no modelo OSI. Ela procura definir um protocolo próprio para cada camada, assim como a interface de comunicação entre duas camadas adjacentes.

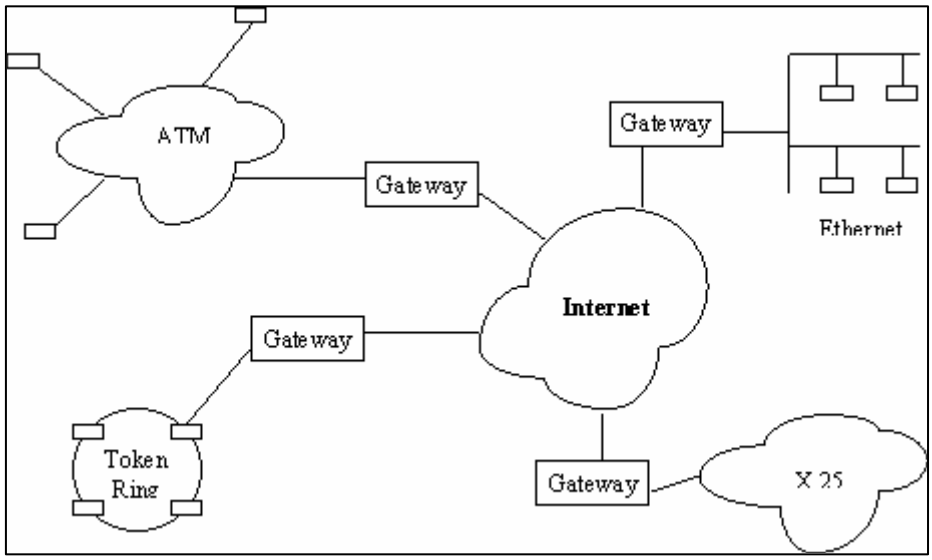
FIGURA 2.2 - MODELO EM CAMADAS DA INTERNET

2.2.1 CAMADA DE REDE DE COMUNICAÇÃO

Na camada de rede de comunicação da Internet, não existe um padrão para a sub-rede de acesso, possibilitando a conexão de qualquer tipo de rede, desde que haja uma interface que compatibilize a tecnologia da rede com o protocolo IP. Desta forma, um número muito grande de tecnologias pode ser utilizado na sub-rede de acesso, *como Ethernet, Token Ring, FDDI, X.25, Frame Relay, ATM*, etc.

Para que todas estas tecnologias possam ser "vistas" pela rede Internet, existe a necessidade de uma conversão de endereçamentos do formato utilizado pela sub-rede e o formato IP. Esta conversão é realizada pelos *gateways*, que tornam a interconexão das redes transparente para o usuário conforme figura 2.2.1. Além das conversões de protocolos, os *gateways* são responsáveis pela função de roteamento das informações entre as sub-redes ([TAN1996]).

FIGURA 2.2.1: INTERCONEXÃO DE SUB-REDES À REDE INTERNET



Fonte: [TAN1996].

2.2.2 CAMADA DE INTER-REDE

A camada de Inter-Rede, também chamada de Internet, é equivalente à camada de rede do modelo OSI. Nela são especificados vários protocolos, dentre os quais se destaca o IP.

Segundo [TAN1996] o IP é um protocolo não orientado a conexão, cuja função é transferir blocos de dados denominados *datagramas* da origem até o destino, podendo passar inclusive por várias sub-redes (a origem e o destino são *hosts* identificados por endereços IP).

A operação no modo *datagrama* é uma comunicação não confiável, não sendo usado nenhum reconhecimento fim a fim ou entre nós intermediários, nem qualquer tipo de controle de fluxo. Nenhum mecanismo de controle de erro de dados é utilizado, apenas um controle de verificação do cabeçalho, para garantir que os *gateways* encaminhem as mensagens corretamente. Algumas das principais características do protocolo IP são as seguintes:

a) Serviço de *datagrama* não confiável;

b) Endereçamento hierárquico;

c) Facilidade de fragmentação e remontagem de pacotes;

d) Campo especial indicando qual o protocolo de transporte a ser utilizado no nível superior;

e) Identificação da importância do *datagrama* e do nível de confiabilidade exigido;

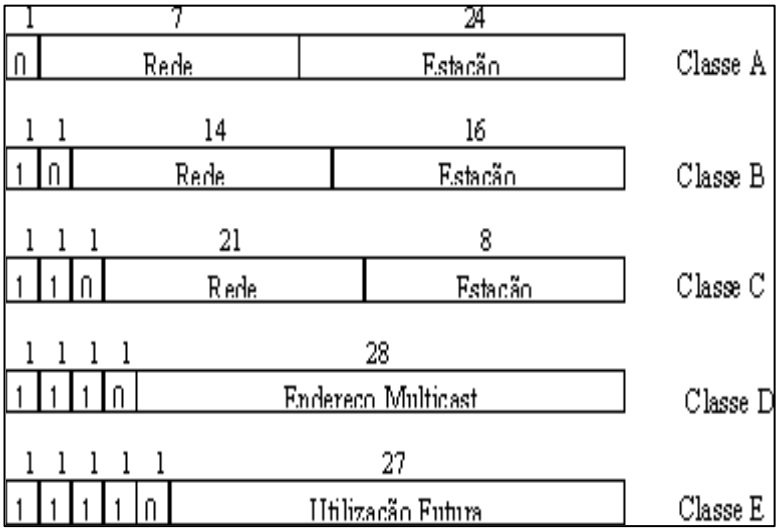
f) Descarte e controle de tempo de vida dos pacotes inter-redes no *gateway*.

2.2.2.1 ENDEREÇAMENTO IP

O roteamento dos datagramas através das sub-redes é feito baseado no seu endereço IP, número de 32 bits normalmente escritos como quatro octetos (em decimal), por exemplo 9.179.12.66. Devido ao fato de existirem redes dos mais variados tamanhos compondo a inter-rede, utiliza-se o conceito de classes de endereçamento.

FIGURA 2.2.2.1: FORMATO DOS ENDEREÇOS IP

Fonte: [TAN1996].



Conforme a figura 2.2.2.1 os endereços IP indicam o número da rede e o número do *host*, sendo que a classe A suporta até 128 redes com 16 milhões de *hosts* cada uma, a classe B 16384 redes com até 64 mil *hosts* cada uma, a classe C 2 milhões de redes com até 256 hosts cada uma e a classe D, onde um datagrama é dirigido a um grupo de *hosts*. Os endereços a partir de 1111 estão reservados para uso futuro.

A Internet utiliza a classe C para endereçamento de suas redes e máquinas. Quando um novo provedor de acesso se conecta à ela, ele recebe 256 endereços para serem utilizados pelos seus *hosts* (ou "usuários"). Como um provedor pode ter mais de 256 clientes, ele utiliza um esquema de alocação dinâmica de IP, ou seja, quando o usuário se conecta ao provedor de acesso, ele recebe um endereço IP, podendo desta forma haver até 256 usuários conectados simultaneamente a um provedor de acesso ([TAN1996]).

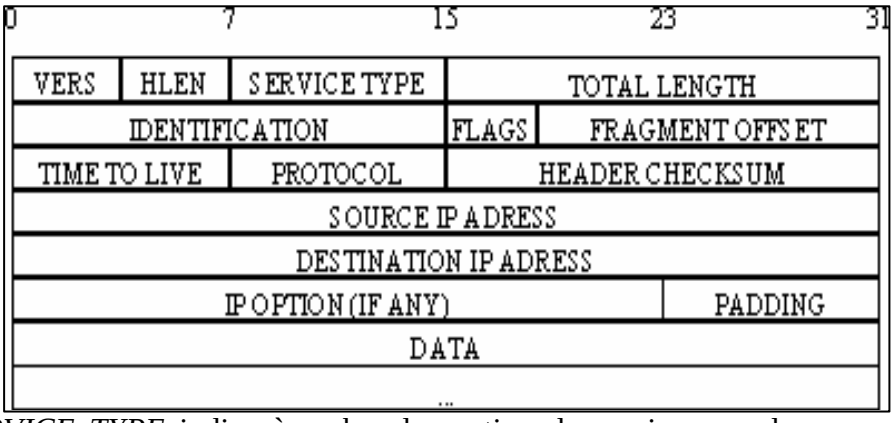
2.2.2.2 FORMATO DO DATAGRAMA IP

O protocolo IP recebe da camada de transporte mensagens divididas em *datagramas* de 64 kbytes cada um, sendo que cada um destes é transmitido através da Internet, sendo

ainda possivelmente fragmentados em unidades menores à medida em que passam por sub-redes. Ao chegarem ao seu destino, são remontados novamente pela camada de transporte, de forma a reconstituir a mensagem original.

O datagrama utilizado pelo protocolo IP consiste em um cabeçalho e um *payload*, sendo que o cabeçalho possui um comprimento fixo de 20 bytes mais um comprimento variável (figura 2.2.2.2).

FIGURA 2.2.2.2: CABEÇALHO IP



Fonte: [TAN1996].

O campo *VERS* identifica a versão do protocolo que montou o quadro. O campo *HLEN* informa o tamanho do quadro em palavras de 32 bits, pois este pode ser variável. O campo *SERVICE TYPE* indica às sub-redes o tipo de serviço que deve ser oferecido ao *datagrama* (por exemplo, para transmissão de voz digitalizada necessita-se mais de uma entrega rápida do que um controle rigoroso de erros, ao passo que para um serviço de transferência de arquivos, o tempo de entrega pode ser sacrificado para se obter um maior controle de erro).

O campo *TOTAL LENGTH* armazena o comprimento total do *datagrama* (dados e cabeçalho), com um valor máximo de 65.536 bytes. O campo *IDENTIFICATION* possibilita ao *host* determinar a que *datagrama* pertence um fragmento recém-chegado (todos os fragmentos de um *datagrama* possuem o mesmo valor). O campo *FLAGS* é composto de um bit não utilizado seguido por dois bits, DF e MF. O DF significa *Don't Fragment* e indica que os *gateways* não devem fragmentar este *datagrama* (por incapacidade do destino juntar novamente os fragmentos). *MF* significa *More Fragments*, e é utilizado como dupla verificação do campo *TOTAL LENGTH*, sendo que todos os fragmentos, menos o último possuem este bit setado.

O *FRAGMENT OFFSET* informa a que posição no *datagrama* atual pertence o fragmento. O campo *TIME TO LIVE* é um contador utilizado para se limitar o tempo de vida de um pacote. Quando o *datagrama* é criado, este campo recebe um valor inicial, que é decrementado toda vez que passa por um *gateway*. Quando este contador atinge o valor zero, isto indica que a rede está em congestionamento ou que o *datagrama* está em *loop*, e o *datagrama* é descartado.

O campo *PROTOCOL* indica o protocolo que gerou o *datagrama* e que deve ser utilizado no destino. O campo *HEADER CHECKSUM* é utilizado pelos *gateways* para se fazer uma verificação do cabeçalho (apenas do cabeçalho, não dos dados), para que o *gateway* não roteie um *datagrama* que chegou com o endereço errado.

SOURCE IP ADDRESS e *DESTINATION IP ADDRESS* são, respectivamente, os endereços de *host* origem e destino. O campo IP *OPTIONS* é usado para o transporte de informações de segurança, roteamento na origem, relatório de erros, depuração, fixação da hora e outras. O campo *PADDING* possui tamanho variável e é utilizado para se garantir que o comprimento do cabeçalho do *datagrama* seja sempre um múltiplo inteiro de 32 bits. Finalmente, o campo *DATA* transporta os dados do *datagrama*.

2.2.2.3 ROTEAMENTO

O roteamento consiste no processo de escolha do caminho através do qual deve ser enviado o dado para o sistema destino. Caso o destino esteja localizado na mesma sub-rede, esta é uma tarefa fácil. Porém quando o destino se encontra em uma sub-rede diferente, a transmissão dos dados é feita através de um *gateway*. O *gateway* faz o roteamento baseado no endereço IP de destino do *datagrama*. Se o *gateway* já estiver conectado à rede para onde o dado deve ser enviado, o problema acabou. Porém, o *gateway* pode não estar ligado diretamente à rede de destino. Neste caso, a partir da identificação da sub-rede, o endereço físico do próximo *gateway* na rota é obtido, através de processos de mapeamento. É importante observar que o *gateway* utilizado em uma rede Internet possui funcionalidades distintas das normalmente aplicadas a ele nas redes OSI.

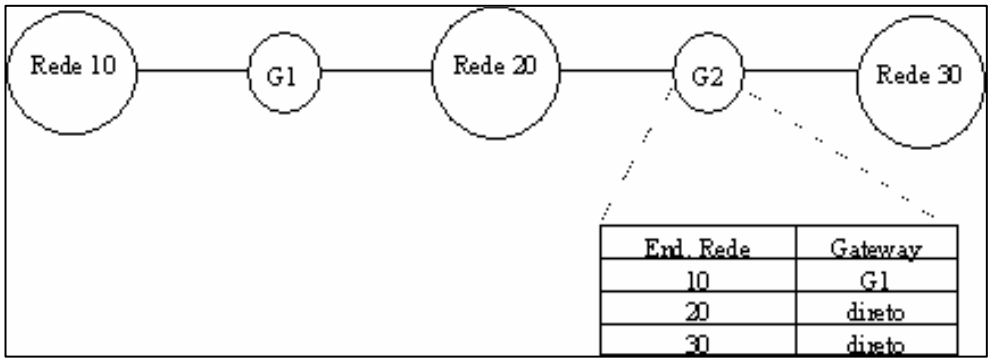
O roteamento pode, então, ser dividido em dois tipos:

a) roteamento direto*:* quando o destino do datagrama se encontra na mesma sub-rede;

b) roteamento indireto*:* quando o destino se encontra em outra sub-rede, necessitando de um *gateway* para o roteamento;

Para realizar o roteamento indireto, os *gateways* utilizam *tabelas de roteamento*, que armazenam informações sobre como atingir cada sub-rede da rede Internet. Uma tabela de roteamento possui, tipicamente, entradas do tipo (N,G), onde N é um endereço IP (de destino) e G é o endereço IP do próximo *gateway* a ser utilizado para se atingir N (figura 2.2.2.3).

FIGURA 2.2.2.3: TABELA DE ROTEAMENTO



Fonte: [TAN1996].

Para diminuir o tamanho das tabelas de roteamento, existem algumas técnicas a serem utilizadas. Por exemplo, pode-se utilizar rotas *default* (pré-estabelecidas) para quando não se encontra referência na tabela sobre uma determinada rota. Este caso se aplica tipicamente para redes que possuem um único *gateway*, como por exemplo, departamentos de uma universidade ligados ao *backbone* por apenas um *gateway*. Ao invés de se ter uma rota para cada sub-rede, utiliza-se a rota *default*.

2.2.3 CAMADA DE TRANSPORTE

A camada de transporte tem o objetivo de prover uma comunicação confiável entre dois processos, estando eles ocorrendo dentro da mesma rede ou não. Ela deve garantir que os dados sejam entregues livres de erros, em seqüência e sem perdas ou duplicação.

Conforme [CAR1994] a Arquitetura Internet especifica dois tipos de protocolos na camada de transporte: o User Datagram Protocol (UDP) e o Transmission Control Protocol (TCP). O UDP é um protocolo não orientado à conexão que pode ser considerado como uma extensão do protocolo IP, e não oferece nenhuma garantia em relação à entrega dos dados ao destino. É um protocolo que não oferece meios que permitem garantir uma transferência confiável de dados, uma vez que não implementa mecanismos de reconhecimento, de seqüenciação nem de controle de fluxo das mensagens de dados trocadas entre dois sistemas.

Os *datagramas* podem, portanto, ser perdidos, duplicados ou entregues fora de ordem ao sistema de destino. A aplicação assume, então a responsabilidade pelo controle de erros dentro do nível de qualidade de serviço que a mesma requer.

Já o protocolo TCP oferece aos seus usuários um serviço de transferência confiável de dados, através da implementação de mecanismos de recuperação de dados perdidos, danificados ou recebidos fora de seqüência, minimizando o atraso na sua transmissão.

A cada fragmento transmitido é incorporado um número de seqüência, de forma a não se perder a ordem dos segmentos a serem juntados para formar o *datagrama*. Existe um mecanismo de reconhecimento para executar essa função que funciona da seguinte forma: o reconhecimento transmitido pelo receptor ao receber o segmento X é o número do próximo segmento que o receptor espera receber (X+1), indicando que já recebeu todos os segmentos anteriores a este. Através da análise dos números de segmento, o receptor pode ordenar os segmentos que chegaram fora de ordem e eliminar os segmentos duplicados. Com base no *checksum* que é adicionado a cada segmento transmitido, os erros de transmissão são tratados e os segmentos danificados são descartados. Existe ainda um controle de fluxo baseado no envio da capacidade de recebimento do receptor, contado a partir do último byte recebido, ao transmissor. Desta forma o transmissor consegue controlar a quantidade de dados que são enviados ao receptor para não haver descarte de segmentos nem necessidade de retransmissão, que ocasionam a queda do desempenho da rede.

Para permitir que vários usuários (processos de aplicação) possam utilizar simultaneamente os serviços do protocolo TCP, foi criado o conceito de porta. Para não haver problemas de identificação de usuários, o identificador da porta é associado ao endereço IP onde a entidade TCP está sendo realizada, definindo assim um *socket*. A associação de portas a processos de aplicação (usuários) é tratada de forma independente por cada entidade TCP. No entanto, processos servidores que são muito utilizados, como FTP, Telnet, etc, são associados a portas fixas, divulgadas aos usuários. Uma conexão é identificada pelo par de *sockets* ligados em suas extremidades. Um *socket* local pode participar de várias conexões diferentes com *sockets* remotos. Uma conexão pode ser utilizada para transportar dados em ambas as direções simultaneamente, ou seja, as conexões TCP são *full-duplex*.

2.2.4 CAMADA DE APLICAÇÃO

As aplicações na arquitetura Internet, ao contrário do que ocorre com as OSI, são implementadas de uma maneira isolada, ou seja, não existe um padrão que defina como deve

ser estruturada uma aplicação. Segundo [CAR1994] cada aplicação possui seu próprio padrão, correspondente a um Request for Comments (RFC).

As aplicações acessam diretamente a camada de transporte, TCP ou UDP, através de chamadas padronizadas de serviços e passagem de parâmetros. Para que uma aplicação seja endereçada em uma rede, a Arquitetura Internet implementa o conceito de portas, que são definidas pelas aplicações e gerenciadas pelo TCP ou UDP.

O número de uma porta, associado ao endereço IP da máquina onde a aplicação em questão está sendo processada, permite a localização de qualquer aplicação no contexto de uma rede Internet.

O nível mais alto desta arquitetura tem por função efetuar a interface com as aplicações que desejam utilizar os meios de comunicação de dados. A arquitetura provê aplicações padronizadas para as principais tarefas do segmento de comunicação de dados. Estas aplicações são:

a) File Transfer Protocol (FTP), permite fazer a transferência interativa de arquivos de uma máquina para outra;

a) Simple Mail Transfer Protocol (SMTP), é a implementação para entrega de correio eletrônico;

b) Post Office Protocol (POP), Protocolo usado por clientes de correio eletrônico para manipulação de arquivos de mensagens em servidores de *mail*.

3 TECNOLOGIAS ENVOLVIDAS NO TRABALHO

O protótipo de software desenvolvido neste trabalho, para distribuição de arquivos recebidos por *e-mail* via Internet, engloba tecnologias as quais serão descritas a seguir.

3.1 POST OFFICE PROTOCOL (POP)

A documentação do RFC1225 descreve sobre a versão 3 do Post Office Protocol (POP3), qual permite uma estação de trabalho acessar dinamicamente uma caixa postal em um servidor *host*. Usualmente, isso significa que POP3 é usado para permitir a estação de trabalho receber *e-mail* do servidor.

Com a evolução da Internet, cada vez mais usuários passaram a comunicar-se via *modem*. Desta forma o SMTP, que pressupõe comunicação *on line* entre os computadores, deixou de ser um protocolo satisfatório para recepção de *e-mails*, pois muitos usuários não acessavam seus *e-mails* diretamente no sistema UNIX que recebia as mensagens, mas de suas residências ou estações remotas. Para solucionar este problema, foi criado o protocolo POP, no qual são criados espaços de armazenamento em servidores para posterior entrega ao usuário. Neste sistema, o endereçamento é feito do mesmo modo que no SMTP, porém a conta destinatária reside num servidor e não diretamente no computador que o usuário utiliza para receber *e-mails*. Para acessar seus *e-mails*, o usuário entra em contato com o servidor que armazena suas mensagens, identifica-se através de uma senha e solicita seus *e-mails*. As mensagens são então transferidas para o computador do usuário, podendo ser apagadas do servidor através de solicitação do usuário ([MIT1997]).

3.1.1 FUNCIONAMENTO DO POP

De acordo com [MIT1997] o servidor *host* inicia o serviço POP3 escutando pelo TCP na porta 110. Quando um cliente *host* deseja fazer uso do serviço, estabelece uma conexão TCP com o servidor *host*.

Quando a conexão é estabelecida, o servidor POP3 envia uma resposta “OK”, informando que está pronto para receber solicitações do cliente. O cliente e o servidor POP3 então trocam comandos e respostas até a conexão ser fechada ou abortada.

Comandos do POP3 consistem de palavras chaves possivelmente seguidos por um argumento. Todos os comandos são terminados por (CRLF), caracteres da tabela ASCII 13 e 10 respectivamente.

Respostas do POP3 consistem de um indicador de sucesso e uma palavra chave seguida possivelmente por informação adicional. Existe atualmente dois indicadores de sucesso: positivo ("+OK ") e negativo ("-ERR"). Respostas para certos comandos tem várias linhas. Depois de enviar a primeira linha da resposta e um CRLF, algumas linhas adicionais são enviadas, cada terminação por um par de CRLF.

A última linha de resposta do servidor é terminado com cinco caracteres CR (Carriage return), LF (line feed), ponto, CR (Carriage return), LF (line feed), ou seja (CFLF.CRLF).

Uma sessão POP3 passa por vários estados durante a conexão. Uma vez a conexão de TCP aberta o servidor POP3 enviará comandos, a sessão entra no estado de *AUTHORIZATION*. Neste estado, o cliente deve identificar-se para o servidor POP3. Uma vez que o cliente tenha feito isso, o servidor adquire recursos associados com a caixa postal do cliente, e a sessão entra no estado de *TRANSACTION*. Neste estado, o cliente requisita ações por parte do servidor POP3. Quando o cliente estiver finalizando suas transações, a sessão entra no estado de *UPDATE*. Neste estado, o servidor POP3 recebe alguns recursos durante o estado de *TRANSACTION* e finaliza. A conexão TCP então é fechada ([MIT1997]).

3.1.1.1 O ESTADO AUTHORIZATION

Uma vez a conexão de TCP foi aberta por um cliente POP3, o servidor POP3 envia uma linha de comando. Este pode ser alguma linha terminado por CRLF. Um exemplo poderia ser:

S. +OK servidor POP3 pronto (Comentários para: eregis@inf.furb.rct-sc.br)

Pode-se observar que esse comando é uma resposta do POP3. O servidor POP3 sempre deveria dar uma resposta positiva como saudação.

A sessão POP3 está agora no estado de *AUTHORIZATION*. O cliente deve enviar agora o comando de *USER*. Se o servidor POP3 responder com um indicador de sucesso positivo (“+OK”), então o cliente pode enviar o comando *PASS* para completar a autorização ou o comando *QUIT* para terminar a sessão POP3.

Se o servidor POP3 responder com um indicador de sucesso negativo ("-ERR") para o comando *USER*, então o cliente pode enviar outro *USER* ou pode enviar o comando *QUIT*.

Quando o cliente enviar o comando de *PASS*, o servidor POP3 usa um par de argumentos do *USER* e *PASS* para determinar se o cliente pode ter acesso a caixa postal apropriada. Nesse caso, o servidor POP3 adquire uma porta de acesso exclusivo na caixa

postal. Se a porta é adquirida com sucesso, o servidor POP3 analisa rigorosamente a caixa postal em mensagens individuais, determina a última mensagem (se existir) presente na caixa postal que foi referenciado pelo comando *RETR*, e responde com um indicador de sucesso positivo. A sessão POP3 entra no estado de *TRANSACTION* agora. Se a porta não poder ser adquirida ou o cliente tiver acesso negado para a caixa postal apropriada, o servidor POP3 responde com um indicador de sucesso negativo. Nesse ponto o cliente dever enviar um novo comando *USER* e iniciar novamente, ou o cliente deve enviar o comando *QUIT*.

Depois que o servidor POP3 analisar as mensagens individualmente na caixa postal, ele coloca uma identificação em cada mensagem. A primeira mensagem na caixa postal é associada a identificação da mensagem como "1", a segunda é 2, e assim por diante.

3.1.1.2 ESTADO DE *TRANSACTION*

Caso o cliente tenha obtido sucesso satisfatório na identificação no servidor POP3 e o servidor POP3 tenha aberto a caixa postal apropriada, a sessão POP3 está agora no estado *TRANSACTION*.

O cliente pode enviar agora repetidamente quaisquer comandos POP3. Depois de cada comando, o servidor POP3 envia uma resposta. Eventualmente, o cliente envia o comando *QUIT* e a sessão POP3 entra no estado de *UPDATE*

Na tabela abaixo é apresentado os principais comandos válidos do POP3 no estado de *TRANSACTION*

TABELA 3.1.1.2 – COMANDOS DO ESTADO TRANSACTION

Comando	Parâmetros	Descrição
Stat	Nenhum	Lista informações da caixa-postal
List	Nº da mensagem	Lista informações da mensagem
Retr	Nº da mensagem	Servidor de e-mail envia a mensagem correspondente para o cliente no formato SMTP
Dele	Nº da mensagem	Marca a mensagem correspondente para deletar
Last	Nenhum	Lista informações da última mensagem postada

3.1.1.3 ESTADO UPDATE

Quando o cliente envia o comando *QUIT* do estado de *TRANSACTION*, a sessão POP3 entra no estado de *UPDATE*. (Observe se o cliente emite o comando *QUIT* do estado de *AUTORIZATION*, a sessão POP3 termina, mas não entra no estado de *UPDATE*).

Quando o estado da conexão estiver em *TRANSACTION* e o cliente enviar o comando *DELE* para o servidor POP3, e logo o comando *QUIT* as mensagem marcadas são deletadas da caixa postal.

3.1.2 FORMATO DA MENSAGEM

O formato das mensagens recebidas pelo protocolo POP3 são definidas conforme RFC822 que refere-se ao protocolo SMTP.

3.2 FILE TRANSFER PROTOCOL (FTP)

De acordo com [HAH1995] o objetivo do *File Transfer Protocol* (FTP) é promover compartilhamento de arquivos.

O FTP é o protocolo de transferência de arquivos da Arquitetura Internet. Trata-se de um utilitário de uso interativo que pode ser chamado por programas para efetuar transferência de arquivos. Os seus principais objetivos são:

a) motivar a utilização de computadores remotos;

b) tornar transparentes ao usuário diferenças existentes entre sistemas de arquivos associados a estações de uma rede;

c) transferir dados de maneira eficiente e confiável entre dois sistemas;

d) promover o compartilhamento de arquivos, sejam programas ou dados.

3.2.1 O MODELO

O FTP trabalha com o modelo Cliente-Servidor. O modelo implementado [RFC959] possui uma característica interessante, que é a de utilizar duas conexões diferentes entre os sistemas envolvidos, uma denominada conexão de controle aos comandos FTP e suas respostas, e a outra denominada conexão de dados à transferência de dados.

A parte executada no cliente (chamada de Cliente-FTP ou Usuário-FTP) pode ser dividida em três módulos que interagem por algum mecanismo interno. Esses módulos são:

a) interface do Usuário;

b) interpretador de Protocolo do Cliente (Cliente- Protocol Interpreter) e

c) processo de Transferência de dados (Cliente- Data Transfer Protocol).

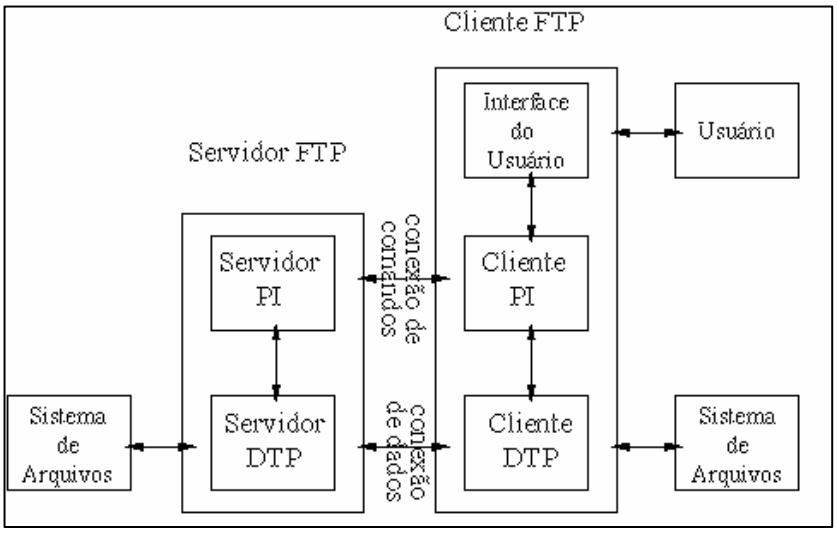
A parte executada no servidor (chamada de Servidor-FTP) é dividida em dois módulos com funções análogas aos seus equivalentes no cliente. Esses módulos são:

a) Servidor- Protocol Interpreter;

b) Servidor- Data Transfer Protocol .

A conexão de controle, usada na transferência de comandos FTP e suas respostas, é realizada diretamente entre o Cliente-PI e o Servidor-PI, e a conexão de dados é estabelecida entre o Cliente-DTP e o Servidor-DTP conforme figura 3.2.1

FIGURA 3.2.1: REPRESENTAÇÃO DE COMUNICAÇÃO FTP CLIENTE E SERVIDOR



Fonte: [CAR1994].

3.2.2 O SISTEMA DE ARQUIVOS

Conforme [CAR1994] cada sistema de arquivo tem suas maneiras de armazenar e acessar seus arquivos, define regras de proteção e de manipulação de maneira diversificada. Dessa forma, torna-se difícil encontrar uma definição comum a todos os sistemas de arquivos existentes.

O FTP, então, define um conjunto de propriedades para os arquivos que podem ser encontradas na maioria dos sistemas, permitindo que um sistema possa manipular um arquivo remoto sem precisar conhecer detalhes sobre o sistema dele, contudo, devem ser definidos parâmetros relativos ao tipo de dados e à estrutura do arquivo

3.2.2.1 TIPO DE DADOS

É interessante que o FTP saiba como os dados estão armazenados nos arquivos, como eles estão representados, para permitir uma melhor adequação dos mesmos no momento de sua transferência.

O FTP, na sua implementação mais completa, suporta quatro (4) tipos diferentes de dados:

a) **ASCII**

O tipo de dados ASCII deve estar disponível em todas as versões do FTP, pois é seu valor padrão. Usa-se preferencialmente na transferência de arquivos-texto, onde um dos sistemas nativos possui uma variação do código ASCII. O sistema que está enviando os dados os converte do seu código interno para o padrão adotado pela arquitetura Internet (o código NVT-ASCII, que identifica o fim de cada linha do arquivo com a seqüência <CRLF>).

Os caracteres são tratados obrigatoriamente como dados de oito (8) *bits*

b) **EBCDIC**

Para facilitar a transferência de dados entre dois sistemas que utilizam o código EBCDIC como código nativo, o código EBCDIC é representado por caracteres de oito (8) *bits*. O código de caracteres é a única diferença entre este código e o ASCII. Mesmo não sendo comum organizar arquivos em linha em EBCDIC, pode-se utilizar o caracter <NL> para indicar o fim de uma linha.

c) **IMAGEM**

Esse tipo de dados é considerado como um conjunto contínuo de *bits* (agrupados em pacotes de 8 *bits* para efeitos de transferência). Se os dois sistemas de arquivos são do mesmo tipo, deve ser garantido que a cópia do arquivo seja idêntica ao original. Isso faz com que esse tipo de dados seja o ideal para transferência de programas executáveis, bibliotecas e outros arquivos que tenham sido compilados.

Este tipo de dado foi criado para facilitar a transferência de dados binários

d) **LOCAL**

O FTP transfere os dados do tipo Local em *bytes* lógicos com tamanho especificado por um segundo parâmetro (agrupados em pacotes de 8 *bits* para efeitos de transferência). O valor deste parâmetro é um número inteiro decimal (não é definido um valor pré-estabelecido).

O destino da transferência possui uma definição interna de qual é o seu tamanho lógico de caractere e faz a conversão para o melhor valor possível de armazenamento.

Esse tipo foi criado para facilitar a transferência de dados binários que dependem do tamanho da unidade de dados, como ponto fixo ou reais de ponto flutuante.

3.2.3 TRANSFERÊNCIA DE ARQUIVOS

De acordo com [CAR1994] para transferir dados deve existir uma conexão de dados entre portas apropriadas e deve ser feita uma escolha de parâmetros de transferência. Os processos Cliente-DTP e Servidor-DTP possuem portas com valores *default* que devem ser suportadas por todas as versões do FTP. Entretanto, o cliente pode alterar o valor de tais portas.

Logo que inicia a transferência de dados, o gerenciamento da conexão de transferência de dados passa a ser responsabilidade do servidor, salvo uma transferência sem erros e em que os dados estão indo do cliente para o servidor. Nesse caso, em vez de enviar um *End of File*, torna-se responsabilidade do cliente fechar a conexão para indicar o fim de arquivo.

Acrescentando às definições existentes do FTP, pode-se definir também, o modo de transferência dos arquivos, de forma a otimizar e melhorar a transferência dos dados. O modo de transmissão pode ser por fluxo contínuo, modo blocado e modo comprimido.

O FTP não se preocupa com a perda ou a adulteração de *bits* durante a transferência, pois é atribuição do TCP - protocolo do nível de transporte, mas provê mecanismos para um eventual reinício da transferência quando ela for interrompida por problemas externos ao sistema (como uma falha na alimentação elétrica).

Este procedimento de reinício só está disponível nos modos de transferência que permitem inserir controles no meio do fluxo de dados (modo de transferência blocado e comprimido).

3.2.3.1 MODOS DE TRANSFERÊNCIA

Além dos parâmetros já descritos para definir característica do arquivo, o FTP possui meios que permitem definir como deve ser realizada a transferência dos arquivos, a fim de otimizar e controlar melhor o processo. O parâmetro que define como o arquivo é transferido, chamado modo de transmissão, pode assumir os seguinte valores:

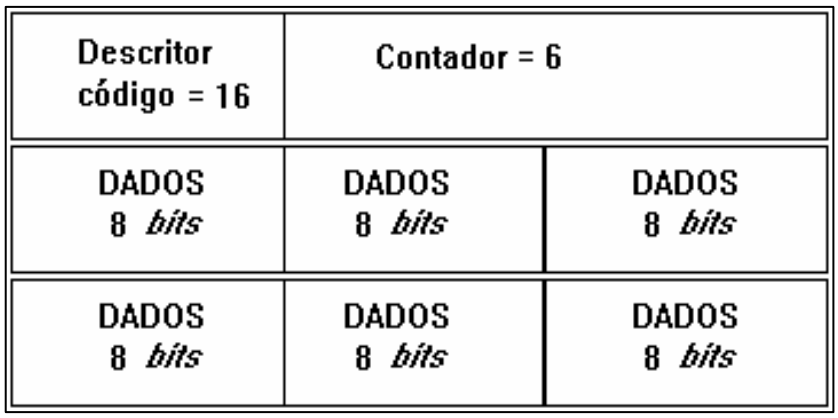
a) **Fluxo contínuo** (*stream*): Os dados são transmitidos como um fluxo contínuo de caracteres. No caso do arquivo ser orientado a registro, são utilizados caracteres de controle para indicar se ocorreu um EOR ou EOF. No caso do arquivo ser não-estruturado, o fim dele é indicado pelo fechamento da conexão de dados. Esse modo não é adequado quando se deseja transferir vários arquivos em uma mesma conexão de dados. Para contornar o problema descrito acima, pode-se alterar as portas usadas pelas conexões no fim de cada transferência ou transferir um arquivo apenas por sessão.

b) **Blocado** (*Block*): O arquivo é transferido como uma série de blocos precedidos por um cabeçalho especial. Este cabeçalho é constituído por um **contador** (2 *bytes*) e um **descritor** (1 *byte*). O contador indica o tamanho do bloco de dados em *bytes*, e o descritor, se este bloco é o último do registro, do arquivo, se é uma marca de reinício ou se ele contém dados suspeitos (com possíveis erros).

Códigos do descritor:
128 - EOR
64 - EOF
32 - erros suspeitos
16 - marcador de recomeço

Exemplificando os códigos acima, para transmitir seis (6) *bytes* o FTP enviaria um (1) *byte* para o descritor - com valor 16, dois (2) *bytes* para o contador - com valor 6, e seis (6) *bytes* de dados; como segue a figura 3.2.3.1

FIGURA 3.2.3.1: FORMATO DO BLOCO



Fonte: [CAR1994].

c) **Comprimido** (*compressed*): O arquivo é transferido utilizando-se uma técnica de compressão de caracteres iguais repetidos. Neste modo de transmissão, são enviados três tipo de informação: dados normais, dados comprimidos e informações de controle. Uma

seqüência de dados comprimidos e informações de controle. Uma seqüência de dados normais é sempre precedida por um campo contendo o tamanho desta seqüência. Os dados comprimidos, por sua vez, são representados por dois campos: o primeiro contém o número de repetições o segundo o caractere a ser repetido.

3.2.4 COMANDOS DO FTP

Os comandos FTP podem ser divididos em quatro (4) grupos:

a) Controle de acesso: São usados para fornecer ao FTP informações sobre as características de quem está realizando a conexão, permitindo o controle de acesso aos arquivos dos sistemas envolvidos.

b) Manipulação de diretórios: Servem para facilitar a manipulação dos diretórios do sistema de destino.

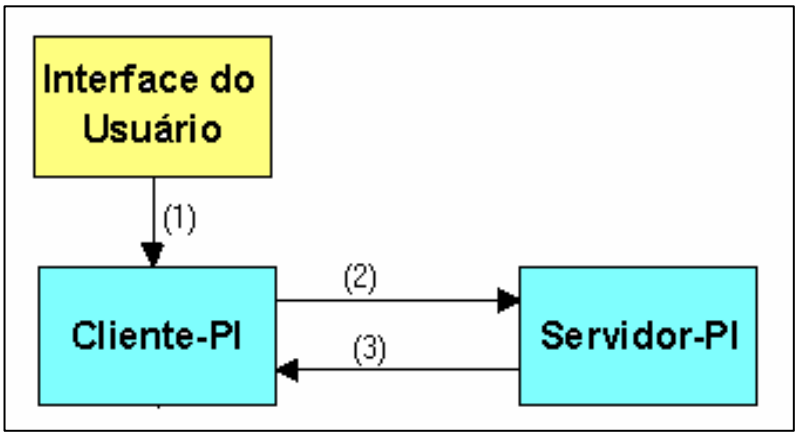
c) Parâmetros da transferência: São usados para especificar os valores dos parâmetros válidos para uma transferência de dados. Eles podem ser especificados em qualquer ordem, mas devem preceder os comandos de serviço.

d) Serviço: Esses comandos são utilizados para solicitar os serviços de transferência ou função do sistema de arquivos. Eles podem ser especificados em qualquer ordem, com exceção do comando *rnfr* (*Rename From*), que deve ser seguido pelo comando *rnto* (*Rename to*), e do comando *rest* (*Restart*), que por sua vez, deve ser seguido por comandos tais como *stor* (*Store*) e *retr* (*Retrieve*).

3.2.5 TRANSFERÊNCIA DE ARQUIVOS

Inicialmente, o usuário chama o utilitário FTP, que dispara o módulo Cliente-FTP, que recebe via interface do usuário um endereço identificando o sistema remoto que se deseja acessar. A partir daí, o Cliente-FTP estabelece uma conexão TCP correspondente à conexão de controle com o sistema remoto desejado associando o Cliente-PI e o Servidor-PI (Figura 3.2.5.1).

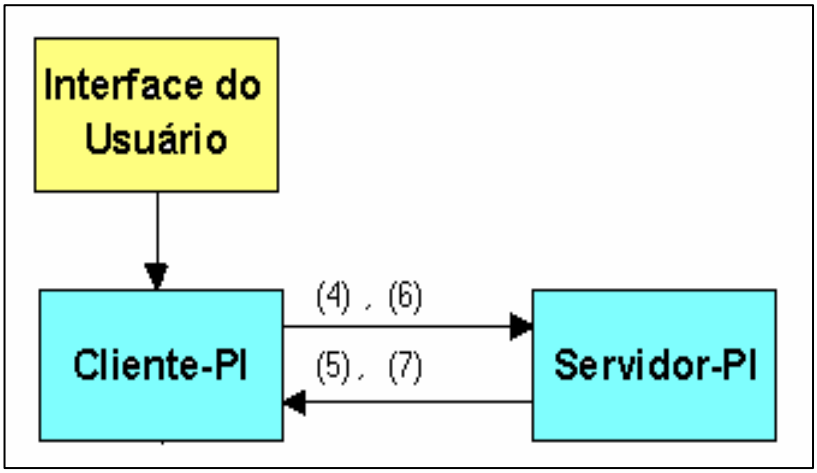
FIGURA 3.2.5.1: ESTABELECIMENTO DA CONEXÃO DE CONTROLE



Fonte: [CAR1994].

Feito isso, o usuário fornece, então o seu nome e senha, a partir dos quais o Cliente-FTP gera os comandos do internos do FTP correspondentes e os envia através da conexão de controle ao Servidor-FTP. O FTP não é responsável pela implementação de mecanismo de controle de acesso, ele apenas passa as informações recebidas ao servidor que, efetivamente, implementa tais mecanismos. Se as informações (nome e senha do usuário) fornecidas forem válidas, é retornada pelo servidor o resposta positiva (Figura 3.2.5.2).

FIGURA 3.2.5.2: SEQUÊNCIA DE LOGIN

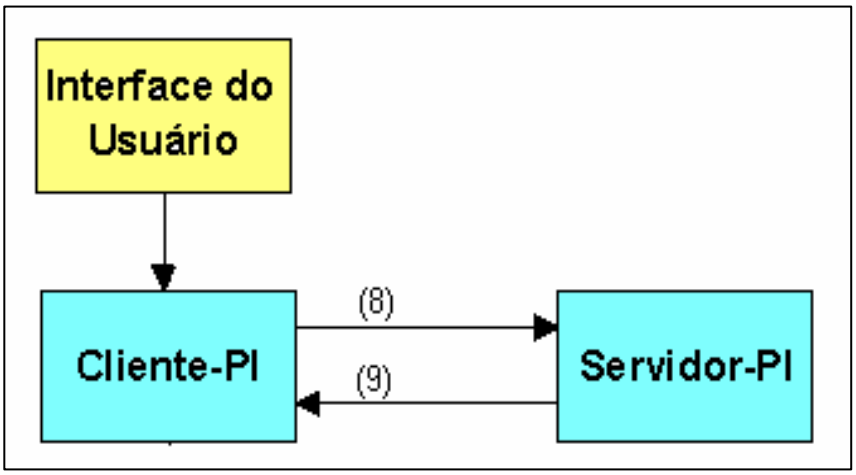


Fonte: [CAR1994].

Nesse ponto o usuário oferece informações que especificam como deve ser efetuada a transferência, tais como o tipo de dado e da estrutura do arquivo a ser transferido e modo de sua transferência. Mais uma vez, o Cliente-FTP obtém tais informações e comandos correspondentes do FTP e os encaminha ao Servidor-FTP via conexão de controle. Quando todos os comandos forem indicados como recebidas e corretos, o Cliente-FTP e os Servidor-

FTP passam a iniciar o processo de transferência do arquivo propriamente dito (Figura 3.2.5.3).

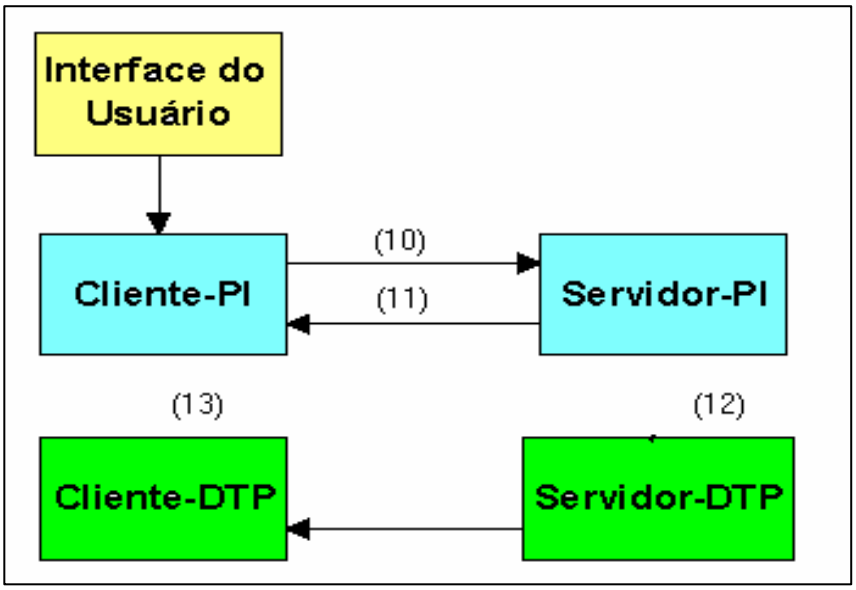
FIGURA 3.2.5.3: ESPECIFICAÇÃO DOS PARÂMETROS DE TRANSFERÊNCIA



Fonte: [CAR1994].

Supondo que a transferência solicitada consista na cópia de um arquivo do Cliente para o Servidor, ela se inicia com a criação dos processos Cliente-DTP e Servidor-DTP e o posterior estabelecimento da conexão de dados entre tais processos (Figura 3.2.5.4).

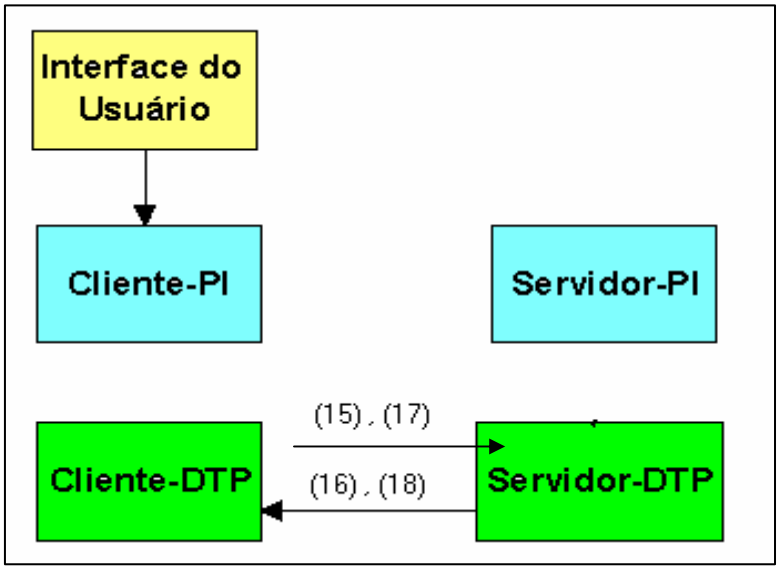
FIGURA 3.2.5.4: ESPECIFICAÇÃO DO COMANDO DE TRANFERÊNCIA E ESTABELECIMENTO DA CONEXÃO DE DADOS



FONTE: [CAR1994].

O cliente e o servidor passam a usar está conexão de dados para que os dados lidos no cliente possam a ser gravados no servidor. Para a transferência dos dados via rede, utilizam-se todos os mecanismos de controle de fluxo e erros que o TCP disponibiliza (3.2.5.5).

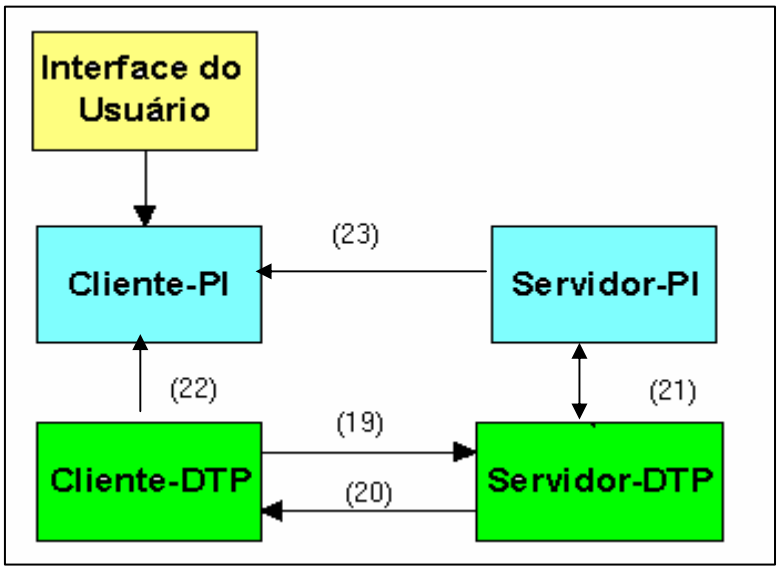
FIGURA 3.2.5.5: TRANSFERÊNCIA DE ARQUIVO



Fonte: [CAR1994].

Ao ser enviado o último caractere do arquivo do destino, o servidor realiza o fechamento da conexão de dados (3.2.5.6).

FIGURA 3.2.5.6: LIBERAÇÃO DA CONEXÃO DE DADOS



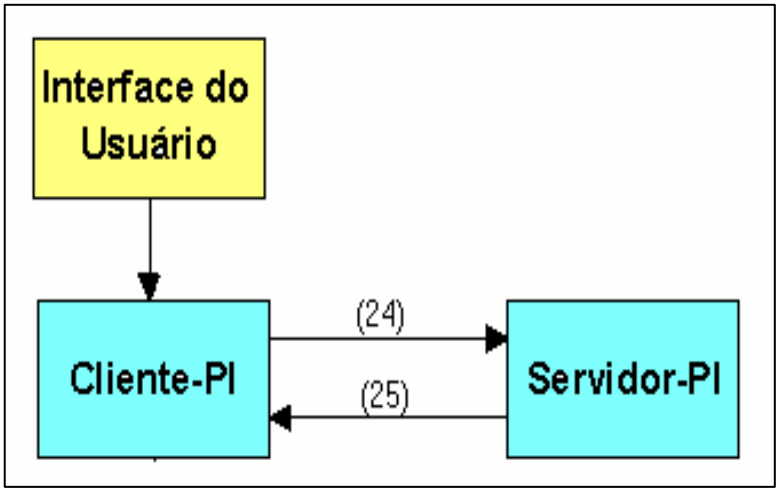
Fonte: [CAR1994].

A conexão de controle permanece aberta para reiniciar todo o processo de transferência quantas vezes for necessário, sendo fechada somente quando o usuário indicar

que sessão do uso do utilitário encerrou. Quando o usuário não quiser realizar mais nenhuma transferência de dados ele solicita o encerramento da sessão (Figura 3.2.5.7).

FIGURA 3.2.5.7: ENCERRAMENTO DA SESSÃO

Fonte: [CAR1994].



3.3 SIMPLE MAIL TRANSFER PROTOCOL (SMTP)

Dentro da concepção tradicional, o correio eletrônico, também conhecido como sistema de mensagens baseado em computador, é uma facilidade que possibilita a usuários utilizando terminais trocar mensagens entre si. Alguns sistemas de correio eletrônico só permitem que as facilidades sejam oferecidas a usuários de um mesmo computador, ao passo que outros permitem que o serviço seja oferecido a usuários de diferentes computadores ligados em rede.

Um exemplo típico dessa última caso de correio eletrônico é o Simple Mail Transfer Protocol (SMTP), que corresponde a um serviços prestados pela camada de aplicação da Arquitetura Internet e costuma estar presente em qualquer implementação.

O sistema de correio eletrônico da arquitetura Internet é implementado pelo SMTP, definido na [RFC822], conforme [CAR1994] o objetivo do SMTP é transferir mensagem com eficiência e segurança entre dois *hosts* independentemente do subsistema de transmissão.

3.3.1 VISÃO GERAL

Com o objetivo de permitir a troca de mensagens através do SMTP, um conjunto de componentes funcionais residentes nos sistemas transmissor e receptor cooperam entre si de forma a garantir o sucesso dessa comunicação

O usuário prepara uma mensagem na sua área de trabalho e a envia a outro usuário do sistema. O ato de enviar um mensagem é uma simples colocação da mensagem em uma caixa postal ou em uma fila de submissão.

A caixa postal é, na realidade, uma entidade associada a um único usuário e administrada pelos sistemas de arquivos, consistindo assim em diretório de arquivos. Tais caixas postais poder estar localizadas um mesmo *host* em *hosts*, diferentes interligados em rede.

Do lado do destinatário, as mensagens entrantes são armazenadas em um arquivo contido no diretório da caixa postal. A partir do seu armazenamento, o usuário tem acesso às mensagens para ler, remover ou mover arquivos pessoais para uso posterior.

3.3.2 FUNÇÕES DO SMTP

As funções do SMTP correspondem aos serviços fornecidos pelo conjunto de comandos do SMTP. Existem algumas funções que são consideradas básicas, tais como as de transmissão de mensagens, que incluem os comandos de *MAIL, RCPT* e *DATA*, e as de abertura e fechamento de um canal de transmissão realizadas através dos comandos *HELO* e *QUIT*. Outras funções são mais específicas e não são oferecidas em todas as implementações do SMTP.

3.3.3 FORMATO DAS MENSAGENS

As mensagens de SMTP possuem formato definido pelo documento [RFC822]. Tal documento apresenta uma série de convenções sem impor restrições muito rígidas sobre a definição do formato das mensagens. Isto acontece porque grande parte das informações nela contidas é orientada para usuários finais, e não para programas.

A estrutura de uma mensagem SMTP é muito simples. Uma mensagem consiste em um cabeçalho de uma ou várias linhas seguido de um texto (corpo).

O cabeçalho é separado do corpo da mensagem através de uma linha em branco.

Uma linha de cabeçalho, por sua vez, consiste em uma palavra-chave seguida pelo caractere “:” e por seus argumentos. Não é permitido que uma linha seja quebrada em várias partes. Como exemplos de palavras-chave, podem-se citar *Received, by, for, Message-ID, From, To, Subject e Date* (Data em que a mensagem foi enviada). A figura 3.2.3 foi utilizado o seguinte cabeçalho e corpo da mensagem.

FIGURA 3.3.3: FORMATO DE UMA MENSAGEM SMTP

Fonte: [CAR1994].

Deve-se observar que todo texto que segue a linha de cabeçalho contendo a palavra-chave "*Cc*" corresponde ao corpo da mensagem.

Como padronização do RFC822 algumas notações utilizadas devem ser esclarecidas:

a) Palavras em letra minúscula representam variáveis ou regras;

b) Aspas (") são utilizadas para delimitar textos literais;

c) Colchetes ([]) são usados para delimitar elementos opcionais;

d) Parênteses são usados para argumento ;

3.4 SOCKET

Por volta de 1980, a Advanced Research Projects Agency (ARPA) fundou um grupo da Universidade da Califórnia no Berkeley para transportar o software TCP/IP para o sistema operacional UNIX e disponibilizar os resultados do software para outros sites. Numa parte do projeto, os desenhistas criaram uma interface que usa as aplicações para comunicação. Eles decidiram usar o sistema UNIX existente chamando quando possível e adicionando novas

chamadas do sistema para suportar funções do TCP/IP que não ajustaram-se facilmente ao conjunto existente de funções.

O resultado foi chamado interface *socket* e o sistema é chamado Berkeley UNIX ou BSD UNIX. A interface *socket é* muitas vezes chamada de Berkeley interface *socket (*[COM1995]).

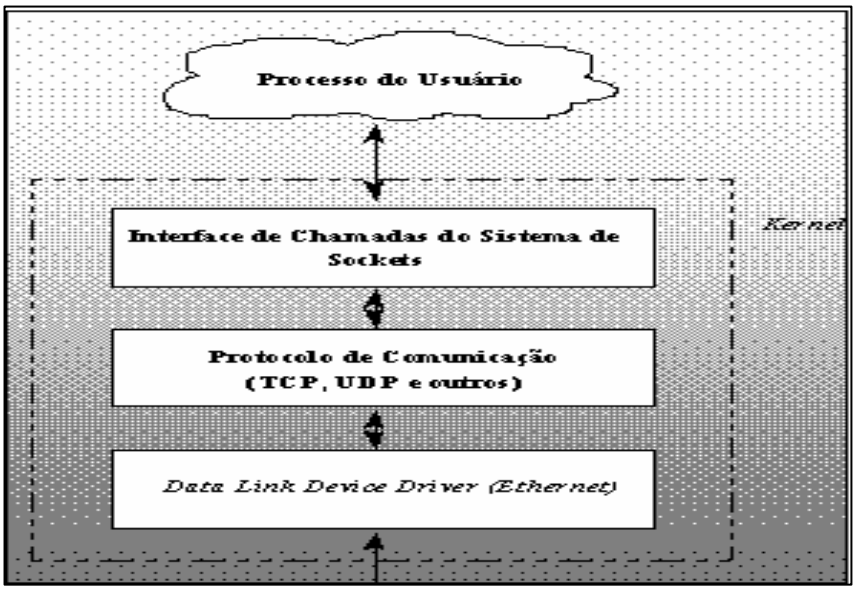
3.4.1 O MECANISMO SOCKET

Os *sockets* são pontos terminais, aos quais podem estar associados as conexões, a partir de sua parte inferior (o lado do sistema operacional), e podem estar associados os processos a partir da sua parte superior (o lado do usuário). A camada do sistema *socket* cria um *socket* (uma estrutura de dados no sistema operacional).

A interface *socket* fornece canais de comunicação *full duplex* e atua como um elemento que recebe as chamadas vindas do processo do usuário e transmite-as ao protocolo de comunicação e este, por sua vez, se encarrega de despachá-la para os níveis inferiores, interface de rede e meio físico, como pode ser observado na figura 3.4.1.

FIGURA 3.4.1: REPRESENTAÇÃO DA INTERFACE *SOCKET* EM UM SISTEMA

Fonte: [TAN1996].



Sockets disponibilizam aos usuários seus serviços através de chamadas de sistema que devem ser adicionadas às aplicações para que possam efetuar a troca de informação com outros processos, locais ou remotos.

A interface *sockets* permite ao usuário desenvolver aplicações que utilizem os serviços orientados à conexão e não orientados à conexão. Sendo que o primeiro garante a confiabilidade e integridade dos dados enviados e o segundo garante uma maior praticidade no envio.

Uma vez que tenha sido criado um *socket*, pode ser alocado espaço em *buffer* para ele, a fim de armazenar os pedidos de conexão que chegam [TAN1996]. A interface é diferenciada pelos diferentes serviços que são produzidos. *Stream, datagram* cada um definindo um diferente serviço disponível para aplicações ([IBM1994]).

A interface *stream socket* define confiança no serviço conexão orientada.

Dados são enviados sem erros ou duplicações e são recebidos na mesma ordem em que foram enviados.

A interface *datagram socket* define um serviço sem conexão. *Datagramas* são enviados como pacotes independentes. O serviço não prove garantias, dados podem ser perdidos ou duplicados, e *datagramas* podem chegar fora de ordem. O tamanho de um *datagrama* é limitado ao tamanho que pode ser enviado em uma transação padrão.

Nenhuma desmontagem e montagem de pacotes é realizada. Um exemplo de aplicação que usas a *datagram sockets* é o *Network File System* (NFS).

Em adicional a função do sistema que implementam *socket*, prevê um conjunto de constantes simbólicas pré-definidas e declaração de estrutura de dados que os programas usam para declarar dados e especificar argumentos. Por exemplo, quando especificado se usar serviço *datagram* ou serviço *stream*, um programa de aplicações usa constantes simbólicas *sock_dgram ou sock_stream.*

3.4.2 CRIAÇÃO DE SOCKET

De acordo com [COM1995] a chamada de sistema *socket* cria os *sockets* solicitados. São necessários três argumentos inteiros e é retornado um resultado inteiro.

Resultado = socket(pf,type,protocol)

O argumento pf especifica a família de protocolos a ser usada com o *socket*, isto é, especifica como interpretar endereços quando eles são fornecidos.

O argumento *type* especifica o tipo de comunicação desejada. Entre os tipos possíveis estão o serviço de transmissão confiável (sock_stream) para *streams e o serviço de transmissão do datagrama sem conexão* (sock_dgram) para *datagramas.*

Para acomodar vários protocolos dentro de uma família, uma chamada a um *socket* requer um terceiro argumento que pode ser usado para selecionar um protocolo específico.

O parâmetro retorna em caso de sucesso um descritor de arquivo usado para referenciar o *socket* criado.

3.4.3 ESPECIFICAÇÃO DE UM ENDEREÇO LOCAL

Inicialmente, é criado um *socket* sem qualquer associação aos endereços local ou de destino. Para os protocolos TCP/IP, isso significa que nenhum número de porta do protocolo local foi especificado. Em muitos casos, os protocolos aplicativos não se preocuparam com o endereço local que usam e são suscetíveis em permitir que o software do protocolo escolha um endereço para eles. Entretanto, os processo do servidor que opera em um porta identificada devem ser capazes de especificar essa porta para o sistema. Quando um *socket* é criado, um servidor usa a chamada de sistema. *Bind* para estabelecer um endereço local para ele. *Bind* possui a seguinte forma:

Bind(socket,localAddr,addrlen);

O descritor do *socket*, utiliza uma estrutura de dados com o endereço e o tamanho dessa estrutura de dados. Retorna o valor 1 em caso de erro.

3.4.4 VINCULAÇÃO DE SOCKETS A ENDEREÇOS DE DESTINO

Inicialmente, um *socket* é criado no estado desconectado, o que significa que ele é associado a qualquer destino exterior. A chamada de sistema *connect* vincula um destino permanente a um *socket*, transferindo para o estado conectado.

Um programa aplicativo deve chamar o recurso *connect* para estabelecer uma conexão antes de transferir dados através de um *socket* de fluxo confiável. Os *sockets* usados com serviços de datagramas sem conexão não precisam ser conectados antes de serem usados, mas essa prática possibilita a transferência de dados sem, a cada vez, especificar o destino.

Connect(socket, destaddr,addrlen)

O descritor do *socket*, ponteiro para estrutura com o endereço destino e o tamanho da estrutura. Essa primitiva deve ser executada por um processo cliente para estabelecer comunicação com o servidor. Após estabelecida a conexão pode haver comunicação entre as duas entidades.

3.4.5 ENVIO DE DADOS ATRAVÉS DE UM SOCKET

Quando um protocolo aplicativo estabelece um *socket*, pode usá-lo para transmitir dados. O argumento *socket* contém um descritor de *socket* em inteiro. O argumento *buffer* contém o endereço dos dados a serem enviados e o argumento *length* especifica o número de bytes a enviar. A chamada *write* fica bloqueada até que os dados possam ser transferidos.

Write(socket,buffer,length)

Retorna inteiro negativo em caso de erro.

3.4.6 RECEBIMENTO DE DADOS ATRAVÉS DE UM SOCKET

A operação *read*, somente poderá ser usada quando o *socket* estiver conectado

Ela possui a seguinte forma:

read(descriptor,buffer,length)

O *descriptor* fornece o descritor inteiro de um descritor de *socket* ou de arquivos dos quais são lidos os dados, *buffer* especifica o endereço da memória onde armazena os dados e *length* especifica o número máximo de bytes a serem lidos.

Retorna o número de bytes recebidos.

3.4.7 ESPECIFICAÇÃO DE UM COMPRIMENTO DE FILA PARA UM SERVIDOR

O servidor cria um socket, vincula-o uma porta de protocolo identificada e aguarda solicitações. Se o servidor usar uma entrega de dados confiável, ou se a resposta ocupar um tempo fora do trivial, alguma solicitação poderá chegar antes que o servidor acabe de responder a uma solicitação anterior. Para evitar rejeições de protocolos ou o descarte de solicitações de entrada, o servidor deve dizer ao software do protocolo básico que ele deseja que tais solicitações estejam em fila até que elas possam ser processadas.

A chamada de sistema *listen* permite que os servidores preparem um *socket* para conexões de entrada. Quando o servidor chama *listen*, também informa ao sistema operacional que o software do protocolo deve enfileirar as várias solicitações simultâneas que chegam ao *socket*. A forma é a seguinte:

listen(socket,qlength)

O argumento *socket* fornece o descritor de *socket* que deve ser preparado para ser usado por um servidor; o argumento *qlength* especifica o comprimento da fila de solicitação para aquele *socket*. Após a chamada, o sistema enfileirará até as solicitações *qlength* para conexões. Se a fila estiver cheia quando uma solicitação chegar, o sistema operacional recusará a conexão descartando a solicitação. *Listen* aplica-se apenas aos *sockets* que selecionaram um serviço de entrega confiável.

3.4.8 COMO UM SERVIDOR ACEITA CONEXÕES

Um processo do servidor usa as chamadas de sistema *socket, bind e listen* para criar um *socket*, vinculá-lo a uma porta de protocolo conhecida e especificar um comprimento de fila para solicitações de conexão. A chamada *bind* associa o *socket* a uma porta de protocolo conhecida, mas o *socket* não é conectado a um destino externo específico. Na realidade o destino deve permitir que o *socket* receba solicitações de conexão de um cliente arbitrário.

Quando um *socket* é estabelecido, o servidor precisa aguardar uma conexão.

Para isso, usa a chamada de sistema *accept*. Uma chamada do recurso *accept* permanece bloqueada até que uma solicitação de conexão chegue. Sua forma é a seguinte:

Accept = accept(socket, addr,addrlen)

No modelo cliente/servidor, faz com que um processo servidor permaneça bloqueado até que uma solicitação de conexão seja feita.
Parâmetros: o descritor do *socket*, um ponteiro para uma estrutura que endereça o solicitante da conexão e um inteiro que informa o tamanho dessa estrutura.

3.4.9 FINALIZAÇÃO DE SOCKETS

Quando um processo acaba de usar um *socket*, ele chama o recurso *close* que possui a seguinte forma:

Close(socket)

O argumento *socket* especifica o descritor de um *socket* a ser fechado. Quando um processo é finalizado por qualquer razão, o sistema fecha todos os *sockets* que permanecerem abertos. Internamente, uma chamada da primitiva *close* decrementa a contagem de *socket* e o destrói se a contagem chegar a zero.

4 TÉCNICAS E FERRAMENTAS UTILIZADAS

Para a realização do protótipo de software apresentado neste trabalho, foi importante estudar algumas técnicas e ferramentas que serviram de apoio à especificação e implementação, visando facilitar seu desenvolvimento e torná-lo mais legível. A seguir são apresentadas algumas considerações sobre as técnicas e ferramentas utilizadas.

4.1 LINGUAGENS GRÁFICAS LIVRES

Na representação do protótipo foram consideras a linguagens gráficas livres, utilizadas para especificação de processos através de fluxogramas.

As linguagens gráficas utilizam uma simbologia gráfica acompanhada de cadeias de símbolos alfanuméricos ou especiais. Têm a vantagem de permitir que um grande número de informações sejam assimiladas rapidamente.

Nas linguagens gráficas livres a simbologia gráfica é arbitrária, ou seja, não está baseada em princípios [MEN1989].

4.2 PROGRAMAÇÃO VISUAL (AMBIENTE DELPHI)

Para implementação do protótipo foi utilizada a linguagem de programação Delphi. Conforme [CAN1996], em uma ferramenta de programação visual como o Delphi, destaca-se o papel do ambiente de programação e interface com o usuário. Possui as características necessárias de orientação a objetos e utiliza a linguagem *Object* Pascal, ou seja, permite ao desenvolvedor programar em linguagens orientadas a objeto. Uma de suas grandes facilidades em programação é poder escrever o código em Delphi, sem saber todos os detalhes da linguagem. Assim sendo, abstrai estes detalhes e torna a implementação mais simples sem a necessidade de conhecimento avançado de programação orientada a objetos.

O suporte a bancos de dados é um considerado um dos recursos mais importantes do ambiente de programação Delphi. A perda de tempo de programadores escrevendo código de acesso a dados é muito grande em muitas linguagens de programação. O Delphi oferece um bom suporte de banco de dados para o desenvolvedor de sistema [CAN1996].

De acordo com [OLI1996], o desenvolvimento de uma aplicação em Delphi ligada a um banco de dados é muito simples. A aplicação é construída utilizando os próprios recursos

do Delphi (componentes), juntamente com outros utilitários. Quando instala-se o ambiente Delphi, são instalados automaticamente drivers para acesso a banco de dados dBASE e o Paradox. Outras informações sobre o Borland Delphi encontram-se em [SWA1996].

4.3 FERRAMENTA CASE (ORACLE DESIGNER 2000)

Para a modelagem das tabelas de configuração dos dados do protótipo foi utilizado a ferramenta Oracle Designer 2000, para apresentar uma visão geral de como as tabelas estão relacionadas. Para maiores detalhes sobre modelagem de entidades podem serem vistas em [SAR1999].

Conforme [FIS1990] e [MAR1996], para reduzir problemas no projeto ou desenvolvimento de um sistema, é relevante a utilização ferramentas CASE. Estas ferramentas, separam o projeto do programa aplicativo da implementação do código, podendo auxiliar a automação de análise, projeto e geração de software.

O Oracle Designer/2000 é uma ferramenta CASE do fabricante Oracle Corporation, que auxilia no processo de desenvolvimento de sistemas, facilitando e agilizando este processo.

O Oracle Disigner/2000 incorpora ferramentas de suporte a modelagem de processos de negócios, análise de sistemas, desenho de softwares e geração de código. Onde os componentes utilizados em casa uma dessas fases geram informações que podem ser aproveitadas pelos componentes das fases seguintes.

4.4 BANCO DE DADOS (PARADOX)

O Banco de dados utilizado para armazenar as informações de configurações do protótipo foi o Paradox.

Para [BOR1995], o Paradox é um banco de dados para o armazenamento e gerenciamento das informações. Possui um sistema de gerenciamento de banco de dados que pode ser utilizado em um computador, como também em um sistema de rede. Uma de suas características é o controle do volume crescente de dados. Outras características é a fácil manipulação das informações armazenadas e fácil entendimento do seu sistema.

O Paradox pode ser considerado como um dos mais rápidos, completos e práticos conjunto de banco de dados relacional. Consiste principalmente em atender às diversas

necessidades dos usuários quanto ao gerenciamento das informações, podendo usá-lo como um sistema, aonde os dados são organizados dentro de tabelas e definidos relações (ou ligações) entre estas tabelas. Deste modo, pode-se extrair ou até mesmo combinar os dados de diversas tabelas, obtendo uma consulta ampla de informações [BOR1995].

4.5 BIBLIOTECA WINSOCK

Para comunicar com os Protocolos de rede foi preciso utilizar a biblioteca Winsock.dll do Sistema Operacional Windows, através da API (Application Programmer Inferface) do Windows pode-se chamar a biblioteca Winsock.dll para desenvolver aplicativos que utilizem *sockets*. Para ter maiores informações do funcionamento da biblioteca *Winsock* pode ser visto em [COM1995].

A biblioteca Windows *Socket* é a adaptação das primitivas *socket* do UNIX BSD para o Microsoft Windows.

O *Winsocket* pode ser considerado mais que uma interface de programação do que propriamente um conjunto de primitivas para manipular o protocolo TCP.

Características da biblioteca *Winsocket*:

a) suporte a vários protocolos, viabilizando inclusive comunicação multimídia;

b) serviços para endereçamento em vários domínios: DNS, SAP, X.500, ...

c) possibilidade de sobreposição das funções send e receive;

d) configuração da qualidade do serviço permitindo variação da largura de banda, tempos de latência, prioridade e agrupamentos de sockets;

e) permite aos usuários utilizarem capacidades multiponto e multicamada específicos do protocolo de transporte disponível;

5 DESENVOLVIMENTO DO PROTÓTIPO DE SOFTWARE

Para o desenvolvimento e implementação do protótipo de software, observou-se como exemplo a necessidade de substituir o tráfego de informações comerciais da rede Pública RENPAC para a Internet.

O serviço de correio da Internet passou a ser avaliada visando redução de custos, atualização tecnológica, rapidez nos processos.

A utilização do recebimento de arquivos recebidos pela Internet podem ser utilizados para atualizar informações num Banco de Dados.

O recebimento desses arquivos exige automatização de alguns processos como:

a) Desanexar os arquivos anexados (*attachment*);

b) Identificar o tipo arquivo;

c) Verificar se o formato está correto;

d) Verificar o caminho e o servidor responsável pelo arquivo;

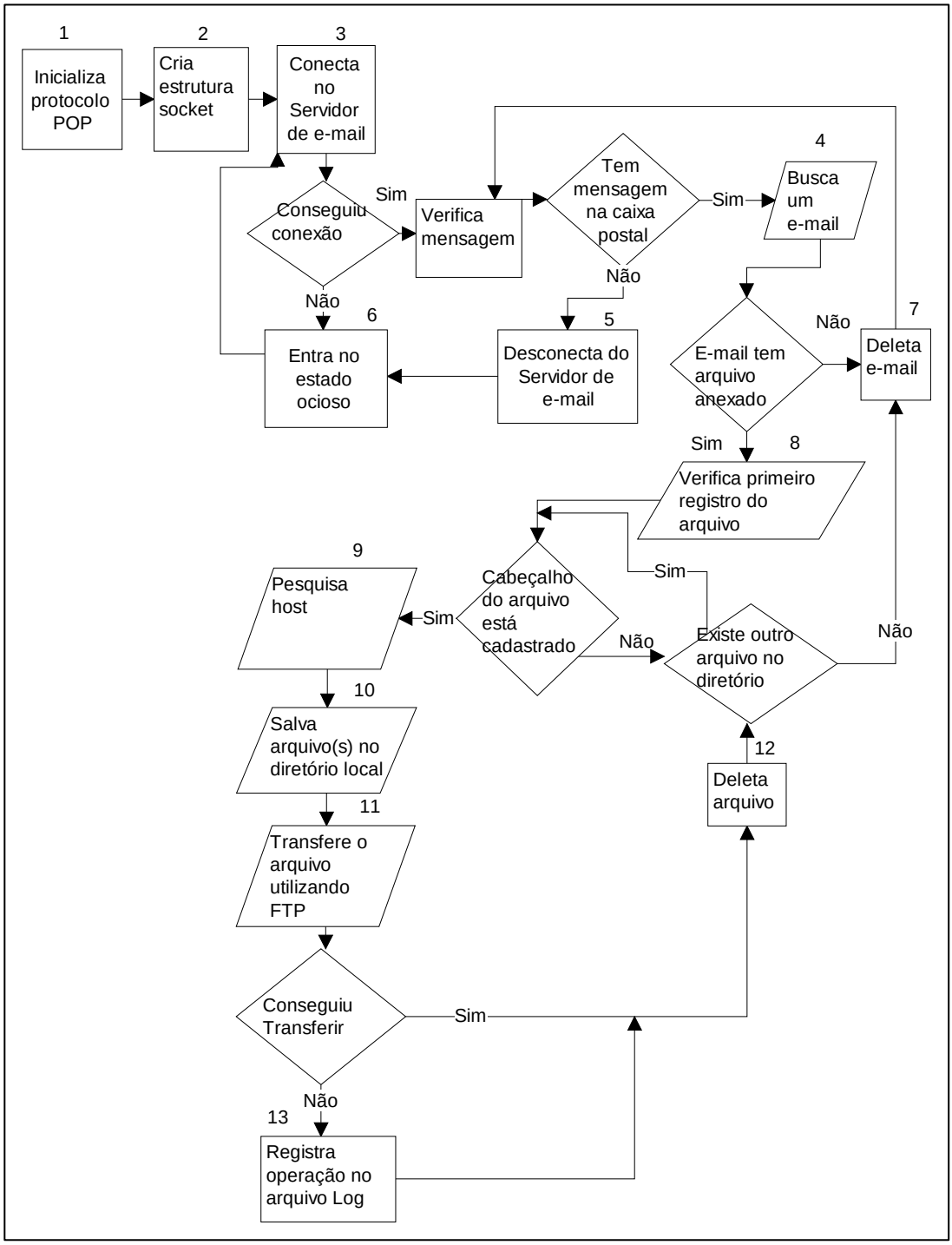
e) Executar o aplicativo FTP para transferir o arquivo;

f) Remover o arquivo da caixa postal;

5.1 ESPECIFICAÇÃO DO PROTÓTIPO

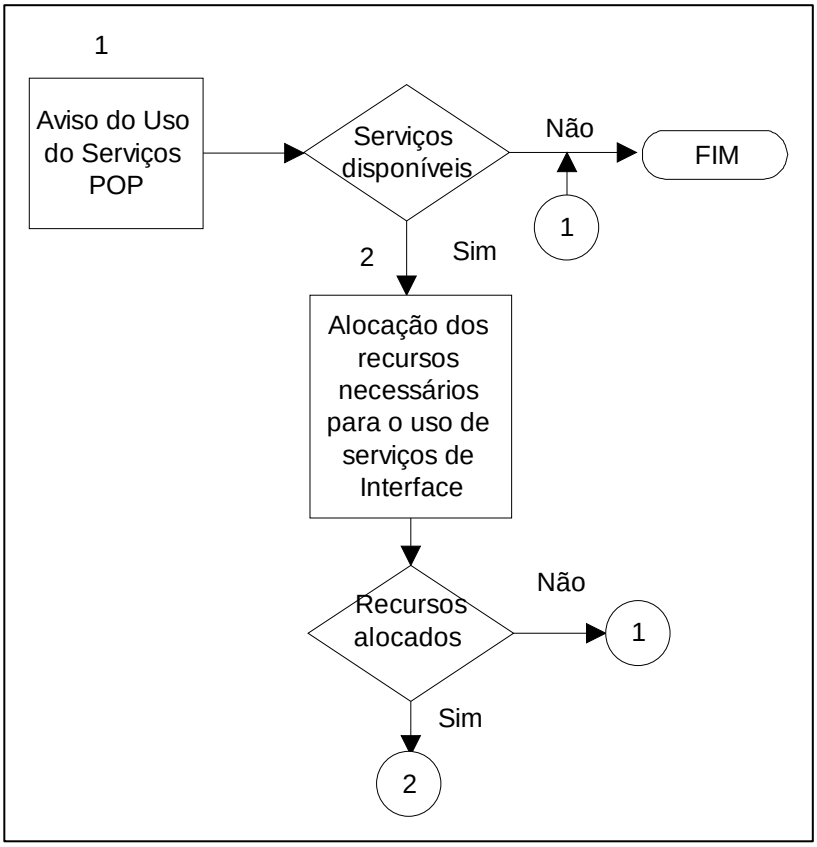
Para diagramar o protótipo foi utilizado a técnica denominada de fluxogramação, conforme apresentado nas figuras a seguir. A figura 5.1.0 apresenta uma especificação genérica do protótipo.

FIGURA 5.1.0: REPRESENTAÇÃO GRÁFICA GENÉRICA DO PROTÓTIPO



A seguir, são apresentados os diagramas correspondentes aos processos do protótipo desenvolvido, com respectivas descrições.

FIGURA 5.1.1: PROCESSO INICIALIZA PROTOCOLO POP

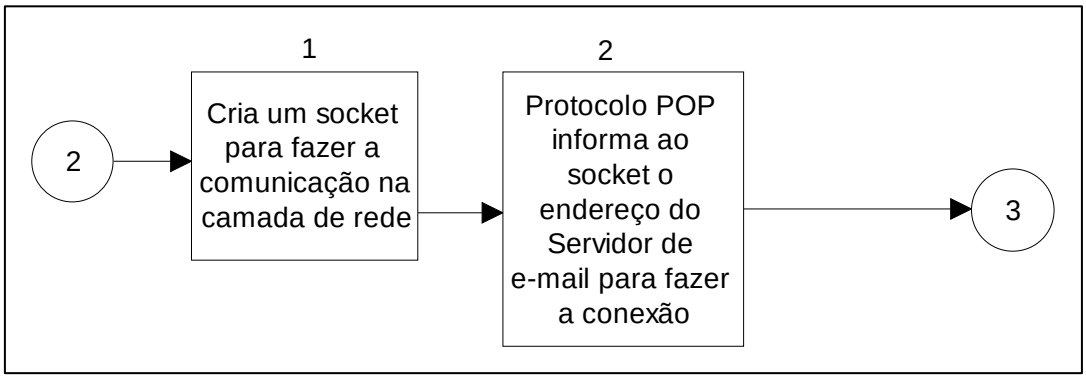


DESCRIÇÃO DO PROCESSO 1: PROCESSO INICIALIZA PROTOCOLO POP (FIGURA 5.1.1)

1: Protótipo avisa a função *socket* que usará o seus serviços POP, habilitando a implementação a realizar quaisquer procedimentos de inicialização e alocação de recursos e retornar alguma informação útil a aplicação.

2: Protótipo aloca e inicializa a memória, recursos, mecanismo de comunicação e estrutura de dados necessários para que a aplicação possa usar os serviços.

FIGURA 5.1.2: PROCESSO CRIA ESTRUTURA *SOCKET*

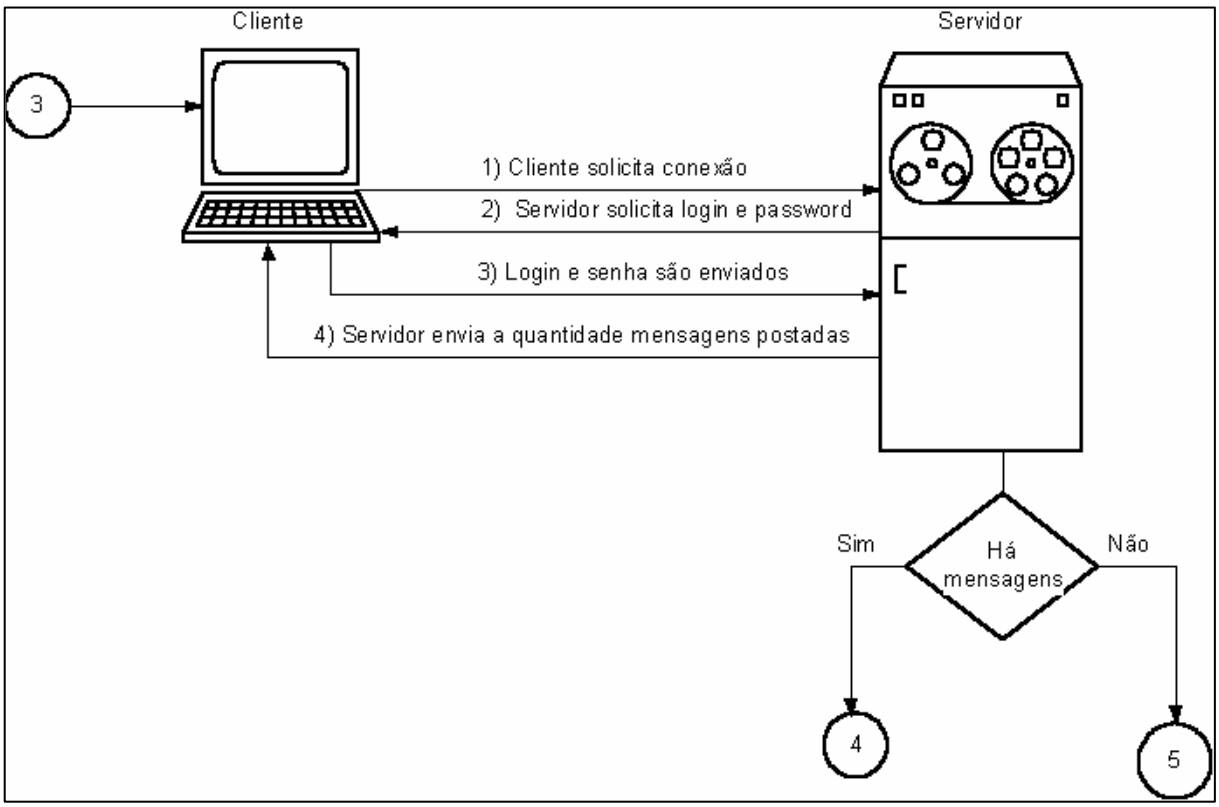


DESCRIÇÃO DO PROCESSO 2: CRIA A ESTRUTURA *SOCKET* (FIGURA 5.1.2)

1: Na criação do *socket* retorna um descritor de arquivo usado para referenciar o *socket* criado.

2: Protocolo POP informa ao descritor do *socket*, que utiliza um ponteiro para estrutura com o endereço destino e o tamanho da estrutura. Essa função deve ser executada por um processo cliente para estabelecer comunicação com o servidor. Após estabelecida a conexão pode haver comunicação entre o protótipo e o Servidor de *e-mail*.

FIGURA 5.1.3: PROCESSO CONECTA NO SERVIDOR DE E-MAIL



DESCRIÇÃO DO PROCESSO 3: CONECTA NO SERVIDOR DE E-MAIL (FIGURA 5.1.3)

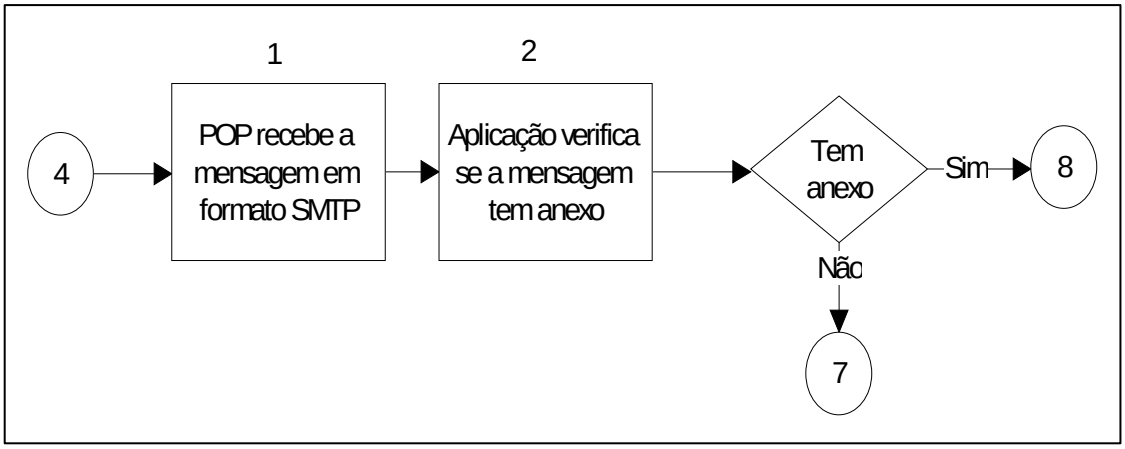
1: Protocolo POP informa ao *socket* a porta 110 para acessar a caixa postal do Servidor de *e-mail*.

2: O servidor envia ao protótipo um comando solicitando o envio do *User* e *password*

3: O protótipo enviar o comando de *USER e PASS*. Se o servidor POP responder com um indicador de sucesso positivo (“+OK”), então está autorizado a ter acesso a caixa postal.

4: O servidor envia a quantidade de mensagens existente na caixa postal.

FIGURA 5.1.4: PROCESSO BUSCA UM E-MAIL

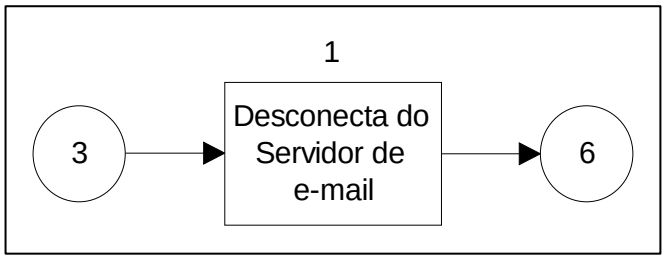


DESCRIÇÃO DO PROCESSO 4: BUSCA *E-MAIL* (FIGURA 5.1.4).

1: O POP recebe a mensagem no formato SMTP.

2: No corpo da mensagem é verificado se existe algum arquivo anexado.

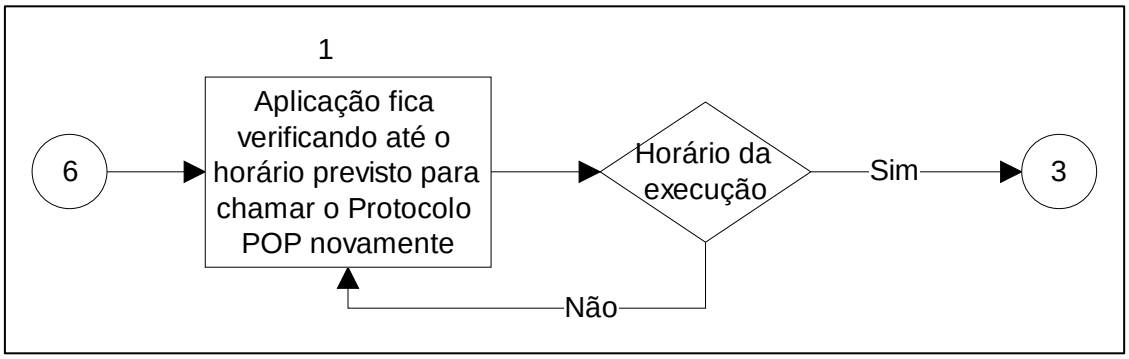
FIGURA 5.1.5: PROCESSO DESCONECTA DO SERVIDOR DE E-MAIL



DESCRIÇÃO DO PROCESSO 5: DESCONECTA A CONEXÃO DO SERVIDOR DE *E-MAIL (*FIGURA 5.1.5)

1: Fecha a conexão com o Servidor de e-mail

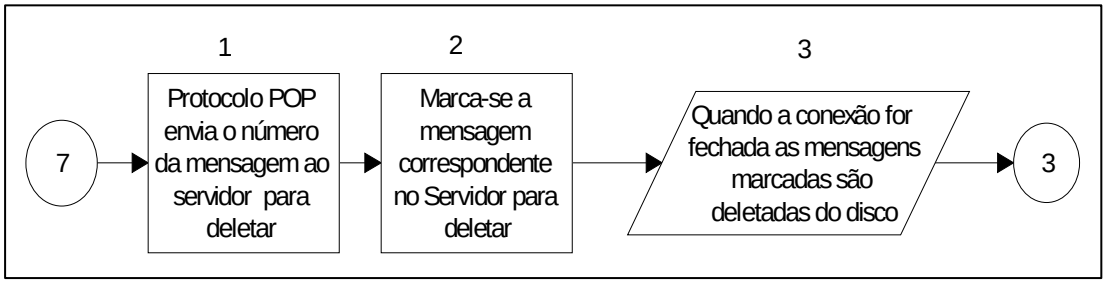
FIGURA 5.1.6: PROCESSO ENTRA NO ESTADO OCIOSO



DESCRIÇÃO DO PROCESSO 6: ENTRA NO ESTADO OCIOSO **(**FIGURA 5.1.6)

1: Aplicação verifica o horário previsto para chamar o Protocolo POP.

FIGURA 5.1.7: PROCESSO PARA DELETAR *E-MAIL*



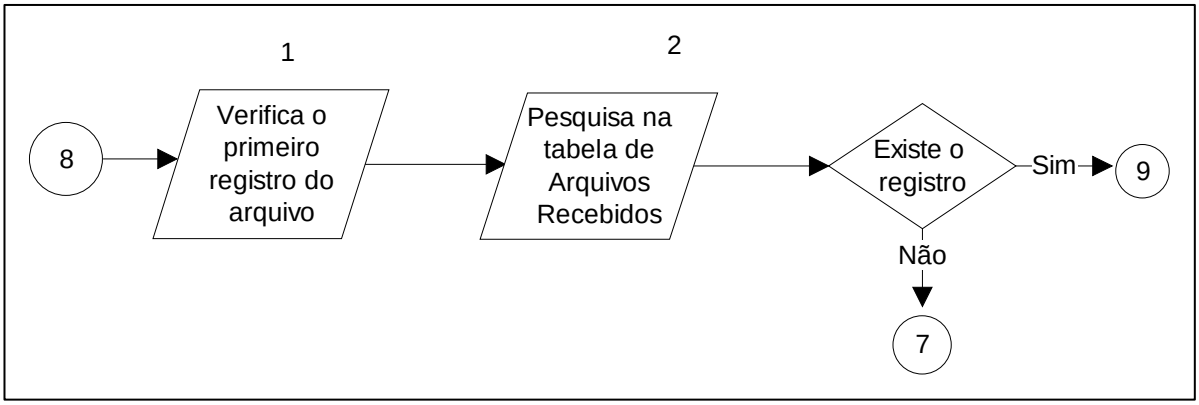
DESCRIÇÃO DO PROCESSO 7: DELETAR *E-MAIL* (FIGURA 5.1.7)

1: O protocolo POP envia o número da mensagem e o comando *dele* para o Servidor de *e-mail*

2: A protocolo POP irá marcar a mensagem que deve ser eliminada e o estado do protocolo deve estar como *TRANSACTION.*

3: Quando o protocolo enviar o comando *quit* o estado do Protocolo passa para *UPDATE e deleta a mensagem.*

FIGURA 5.1.8: PROCESSO VERIFICA PRIMEIRO REGISTRO DO ARQUIVO

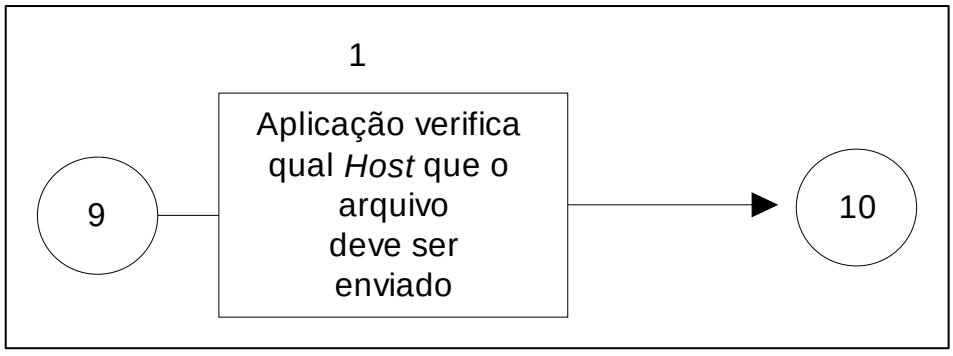


DESCRIÇÃO DO PROCESSO 8: VERIFICA PRIMEIRO REGISTRO DO ARQUIVO (FIGURA 5.1.8)

1: Ao recebimento da mensagem a aplicação verifica se existe arquivos anexados (*attachament*) e o nome do arquivo (*Filename)*

2: O primeiro registro do arquivo é passado para aplicação verificar se na tabela de Arquivo recebidos existe esse registro.

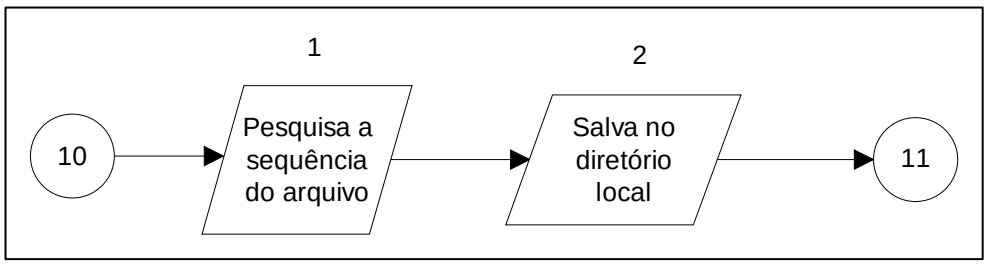
FIGURA 5.1.9: PROCESSO PESQUISA *HOST*



DESCRIÇÃO DO PROCESSO 9: PESQUISA *HOST (*FIGURA 5.1.9)

1:A tabela Arquivos Recebidos(tabela filho) está relacionada com a tabela Servidor FTP(tabela pai) , esse relacionamento existe para que o tipo de arquivo recebido tenha um *Host.*

FIGURA 5.2.0: PROCESSO SALVA ARQUIVO NO DIRETÓRIO LOCAL

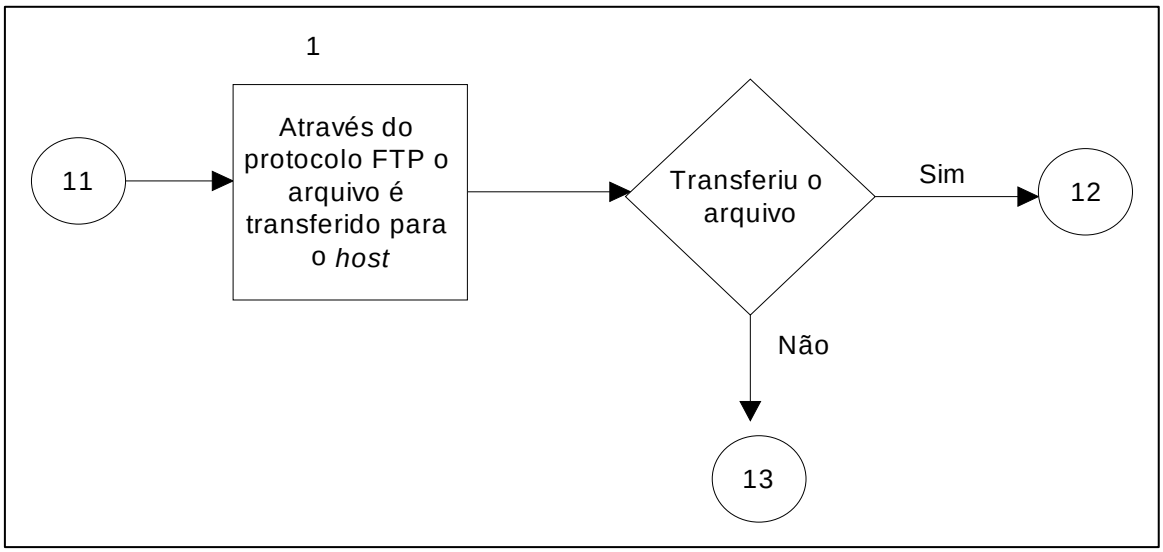


DESCRIÇÃO DO PROCESSO 10: SALVA ARQUIVO NO DIRETÓRIO LOCAL (FIGURA 5.2.0)

1: Pesquisa a seqüência do arquivo na tabela Arquivos recebidos e incrementa a seqüência, em seguida concatena a cinco primeiras posições do nome do arquivo com seqüência do arquivo.

2: Salva o arquivo no diretório local

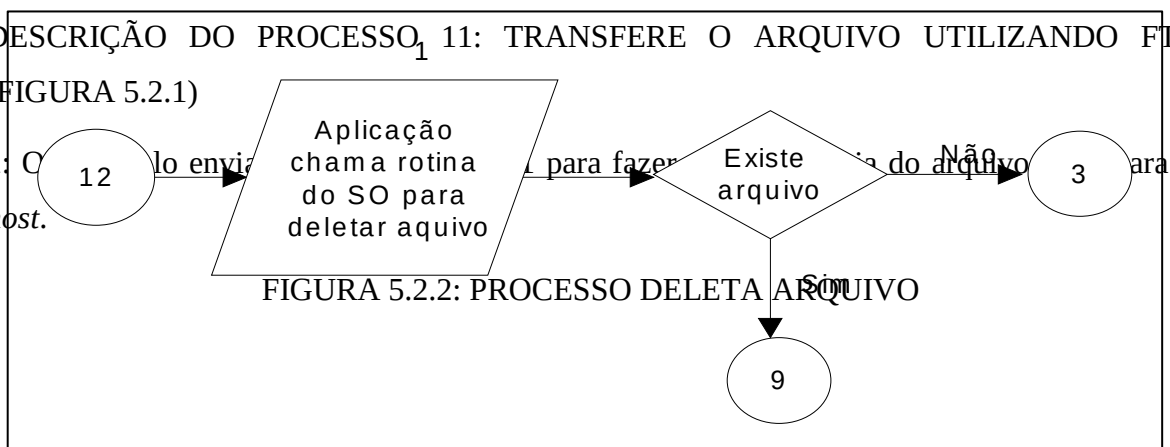
FIGURA 5.2.1: PROCESSO TRANSFERE O ARQUIVO UTILIZANDO FTP



DESCRIÇÃO DO PROCESSO 11: TRANSFERE O ARQUIVO UTILIZANDO FTP (FIGURA 5.2.1)

1: O [illegible] envia [illegible] para fazer [illegible] do arquivo para o *host*.

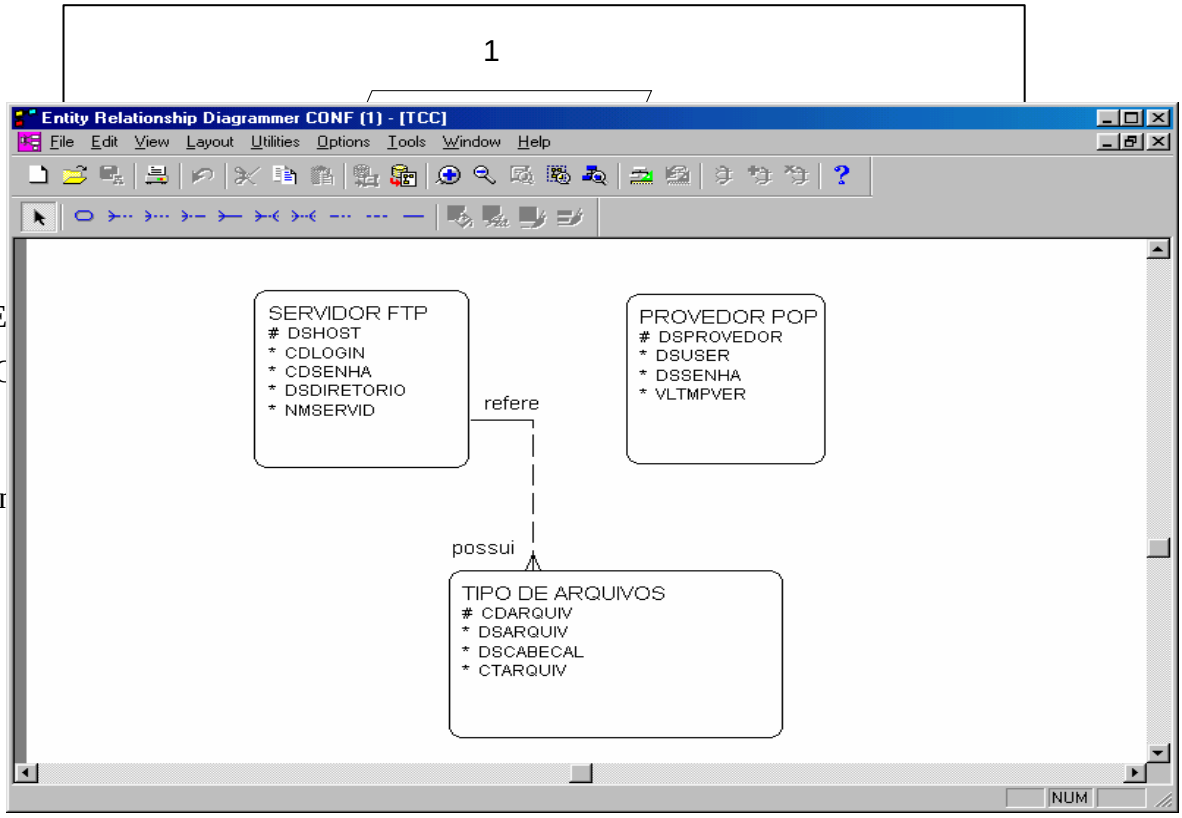
FIGURA 5.2.2: PROCESSO DELETA ARQUIVO



DESCRIÇÃO DO PROCESSO 12: DELETA ARQUIVO (FIGURA 5.2.2)

1: Aplicação chama rotina do Sistema Operacional para deletar arquivo do diretório local.

FIGURA 5.2.3: PROCESSO REGISTRA OPERAÇÃO NO ARQUIVO LOG



DE

FIC

1:

trar

5.2 MODELAGEM DAS ENTIDADES DO PROTÓTIPO

Na figura 5.2.4 apresenta o relacionamento das entidades que servirão como base para construção das tabelas do Protótipo. As entidades servem para dar uma visão geral de como as tabelas estão relacionadas.

FIGURA 5.2.4: RELACIONAMENTO DAS TABELAS DO PROTÓTIPO

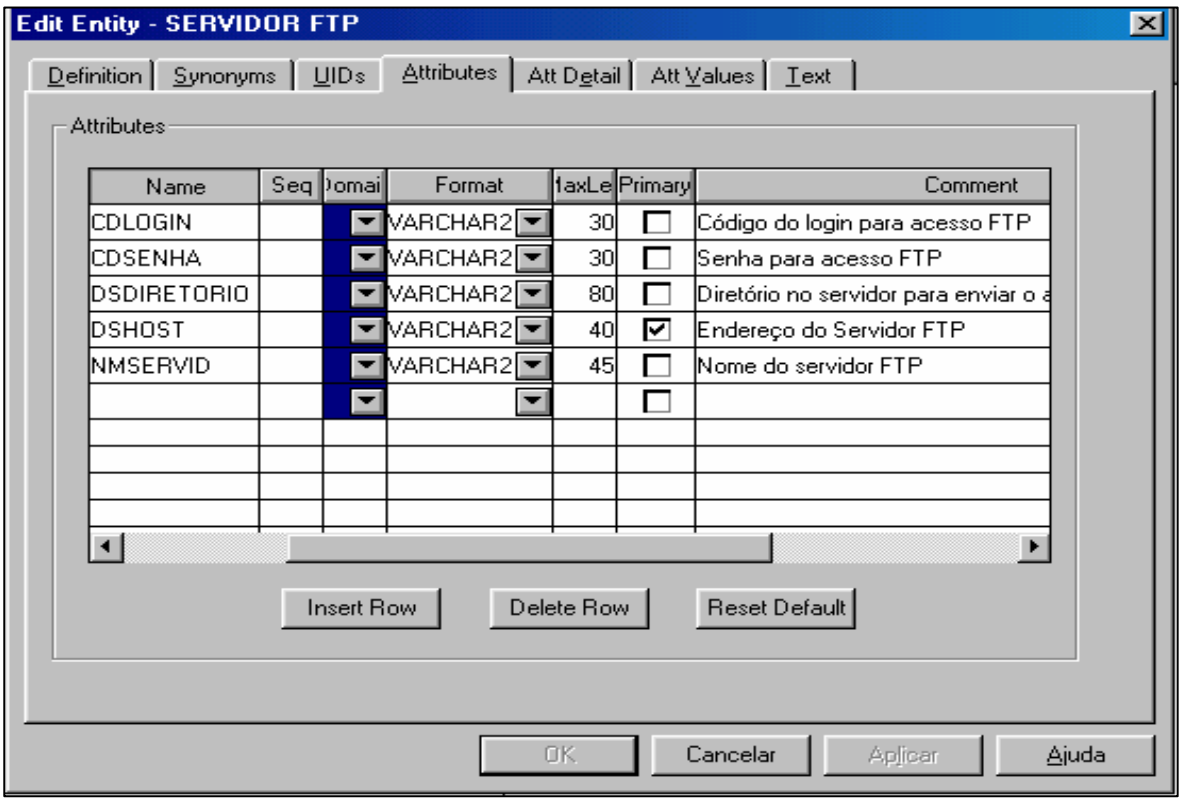


FIGURA 5.2.5: ATRIBUTOS DA ENTIDADE SERVIDOR FTP

FIGURA 5.2.6: ATRIBUTOS DA ENTIDADE TIPO DE ARQUIVOS

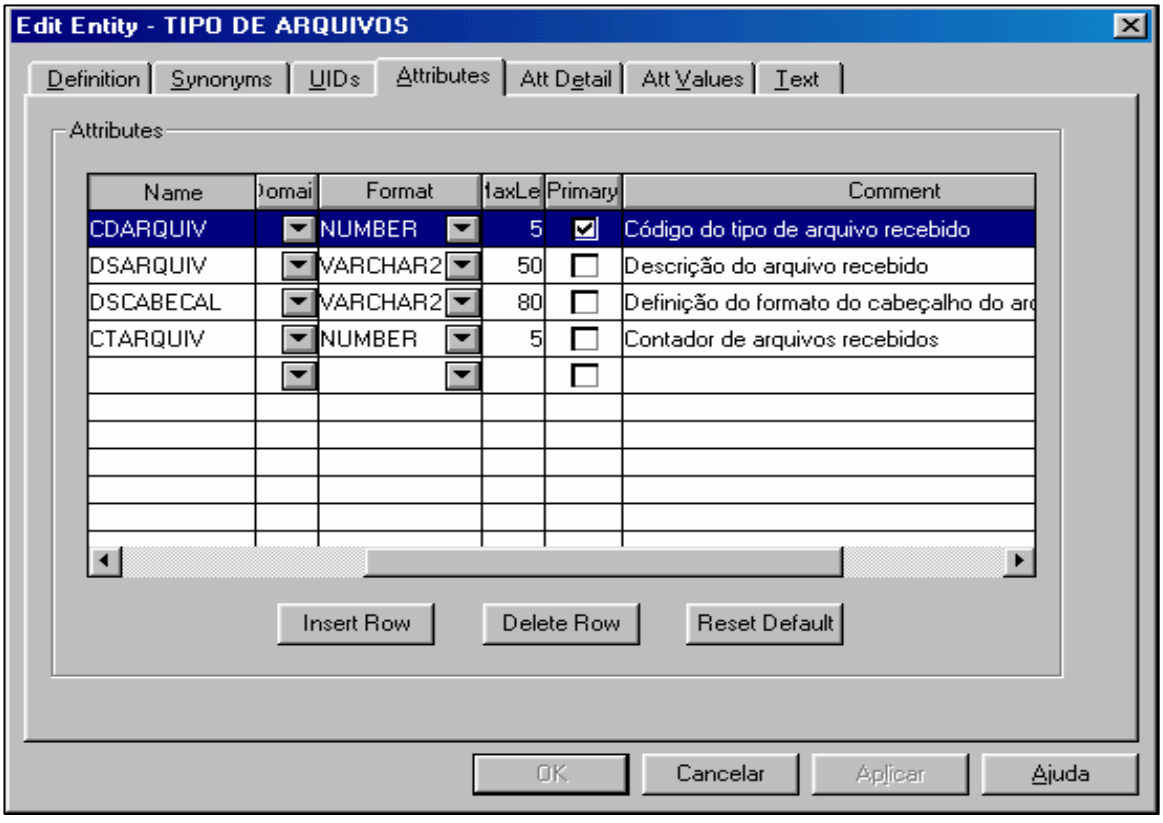
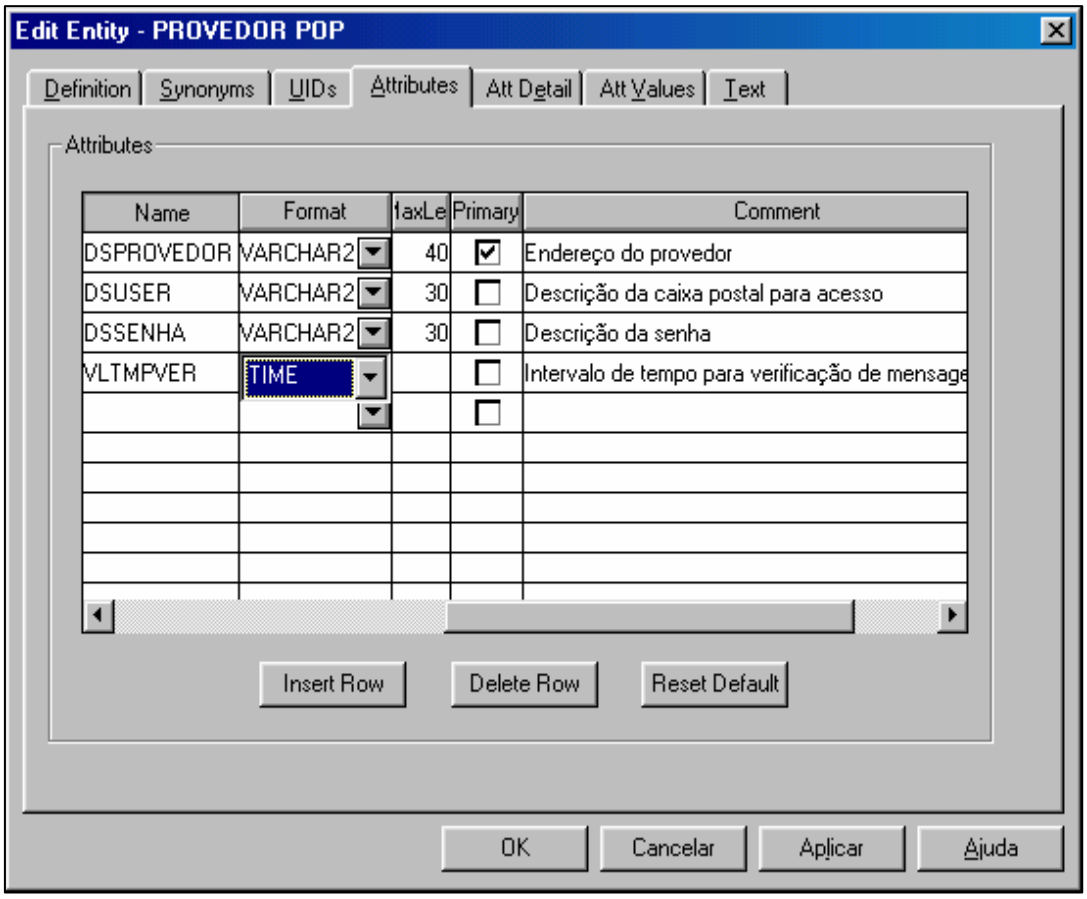


FIGURA 5.2.7: ATRIBUTOS DA ENTIDADE PROVEDOR POP



5.3 IMPLEMENTAÇÃO DO PROTÓTIPO

Para a implementação do protótipo utilizou-se do ambiente de desenvolvimento *Delphi 3.0* para *Windows*.

Por ser descendente do Pascal, linguagem conhecida pelo autor deste trabalho, e também por permitir a utilização de funções que interagem com DLLs, foi feita a escolha do Delphi para a implementação do protótipo.

O processo 3 para conectar no Servidor de *e-mail*, é apresentado os códigos fontes nas linhas abaixo:

PROCESSO 3: FONTE DO PROGRAMA PARA CONECTAR NO SERVIDOR DE E-MAIL

```
procedure TCustomPop3Cli.Connect;
begin
    CheckReady;
    FRequestDoneFlag  := FALSE;
    FReceiveLen       := 0;
    FRequestResult    := 0;
    StateChange(pop3DnsLookup);//Verifica    o    estado    do
protocolo
    FWSocket.OnDataSent      := nil;
    FWSocket.OnDnsLookupDone := WSocketDnsLookupDone;
    FWSocket.DnsLookup(FHost);  //Passa    o  nome  do  servidor
para o socket fazer conexão
end;
```

O processo 4 para buscar um *e-mail* no Servidor de *e-mail*, é apresentado os códigos fontes nas linhas abaixo:

PROCESSO 4: FONTE DO PROGRAMA PARA BUSCAR UM *E-MAIL*

```
procedure TF_Mensag.ExecRecebeArq;
begin
  pvstrarq.Clear;
  if pbnrmsg[1] in ['1'..'9'] then
  begin
    f_pop.Pop3Client.MsgNum  := strtoint(pbnrmsg);
    f_pop.Pop3Client.Retr;
end;
end;
{Inicia Transação}
procedure TCustomPop3Cli.Retr;
begin
    FFctPrv := pop3FctRetr;
//Comando RETR  serve para buscar a mensagem e o número da mensagem que deseja receber
    StartTransaction('RETR',   IntToStr(FMsgNum),
                     pop3Retr, pop3Transaction, RetrDone);
end;
```

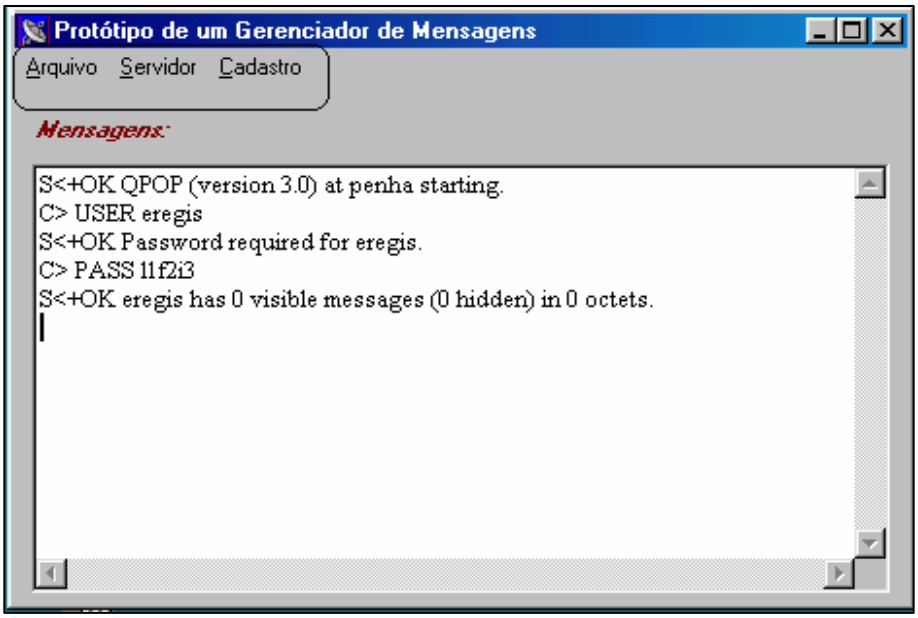
```
procedure TCustomPop3Cli.WSocketDataAvailable(Sender: TObject;
Error: Word);var
    Len  : Integer;
    I, J : Integer;
begin
{Para buscar os dados no Servidor POP, é informado o endereço
da estrutura e o
tamanho maximo (geralmente 4095) do buffer para buscar as
informações}
    Len := FWSocket.Receive(@FReceiveBuffer[FReceiveLen],
                            sizeof(FReceiveBuffer)           -
FReceiveLen - 1);
    //A variável FReceiveBuffer recebe dados do socket
    if Len <= 0 then
        Exit;
    FReceiveBuffer[FReceiveLen + Len] := #0;
    FReceiveLen := FReceiveLen + Len;
    while FReceiveLen > 0 do begin
        I := Pos(#13#10, FReceiveBuffer);
        if I <= 0 then
            break;
        if I > FReceiveLen then
            break;
        FLastResponse := Copy(FReceiveBuffer, 1, I - 1);
        TriggerResponse(FLastResponse);
end;
```

5.4 FUNCIONAMENTO DO PROTÓTIPO

A seguir, será apresentado o funcionamento da implementação do protótipo de software. Nesta apresentação, serão mostradas as interfaces do sistema, bem como a característica de cada uma delas.

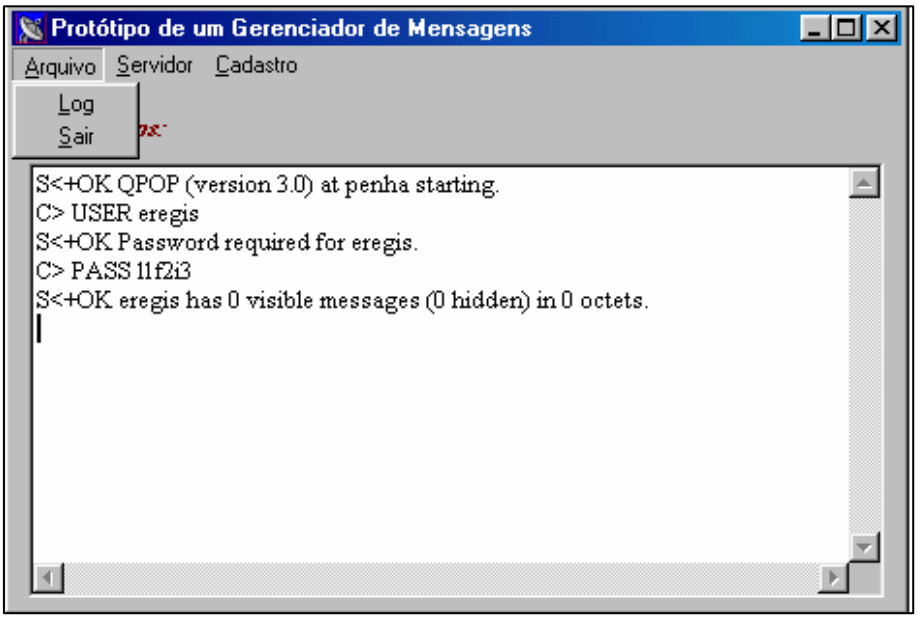
O protótipo de software, possui uma interface de menus para o usuário, conforme mostrado na figura 5.4.1. Essa interface inicial, serve para o usuário definir qual a sua escolha das opções apresentadas e clicar sobre uma delas. O primeiro menu do sistema (Arquivo), tem por objetivo realizar operações com arquivo como o Log e o Sair do protótipo. O próximo menu (Servidor) permite conectar e desconectar com um servidor POP e também executar o protótipo para verificar a existência de mensagens. O último menu do sistema (Cadastro) serve para manter informações dos Servidores POP e FTP e também os tipos de arquivos recebidos.

FIGURA 5.4.1: MENU DE OPÇÕES



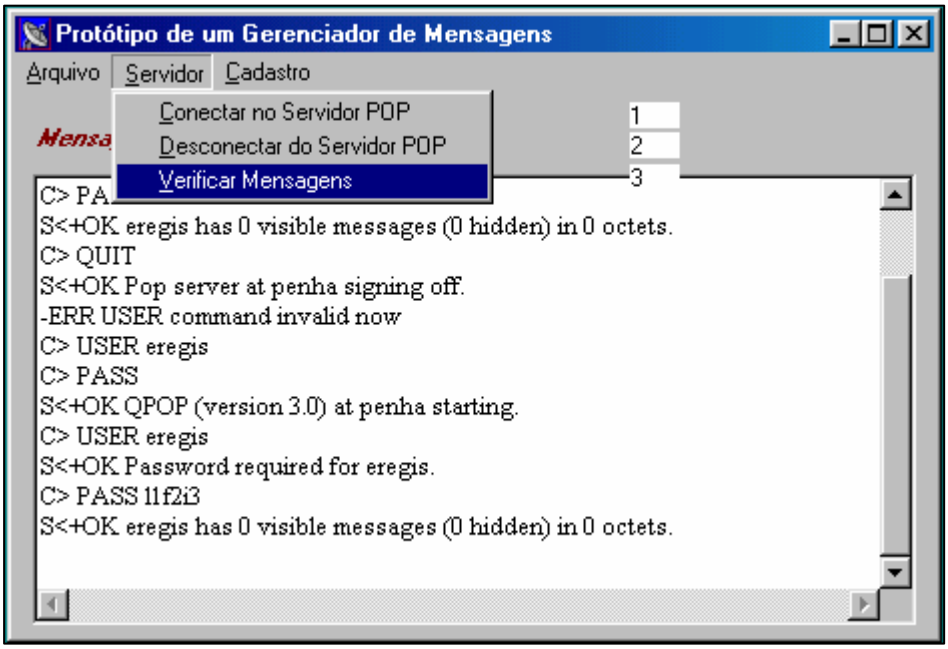
Ao entrar na opção Arquivo, existe a opção Log conforme mostrado na figura 5.4.2. Esta opção serve para abrir o arquivo log. Nesse arquivo ficam registradas as principais atividades do protótipo como conexão, transferência de arquivo e recepção de mensagens. Essa opção serve para o usuário rastrear algum erro, caso houver alguma anomalia na operação. Outra função do menu Arquivo é opção Sair, que fecha a interface principal do protótipo e finaliza a conexão com o Servidor de *e-mail*.

FIGURA 5.4.2: MENU ARQUIVO



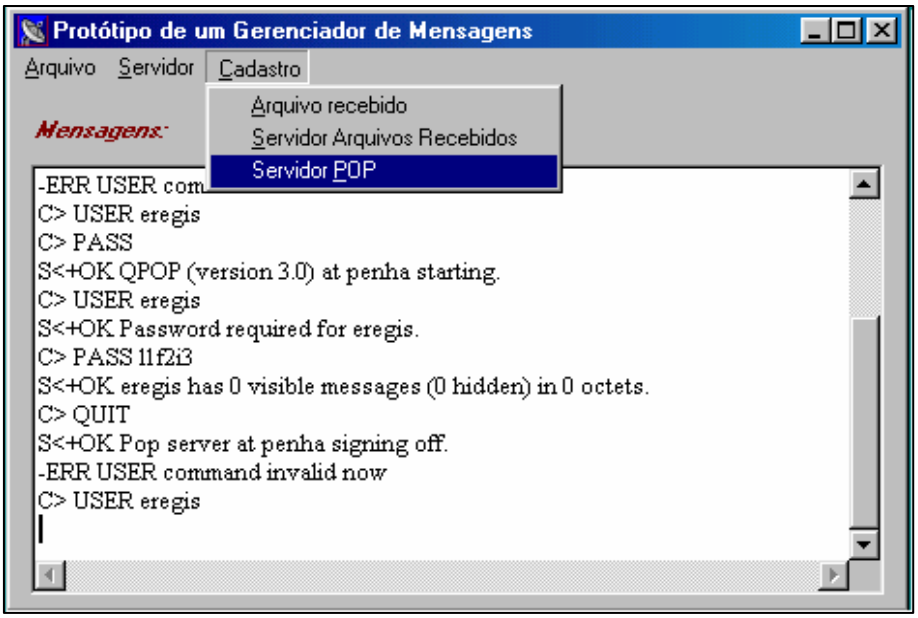
Seguindo para a outra opção de menu (Servidor) conforme figura 5.4.3, encontram-se as três funções (Conectar, Desconectar Servidor POP e Verificar mensagens) que podem ser executadas bastando o usuário clicar sobre uma delas. A função Conectar no Servidor POP (parte 1) permite ao usuário fazer o teste de conexão no Servidor de *e-mail*, a outra função Desconetar fecha a conexão com Servidor POP. A última função Verificar Mensagens (parte 3) permite ao usuário buscar as mensagens no momento, sem precisar esperar o tempo para execução automática.

FIGURA 5.4.3: MENU SERVIDOR



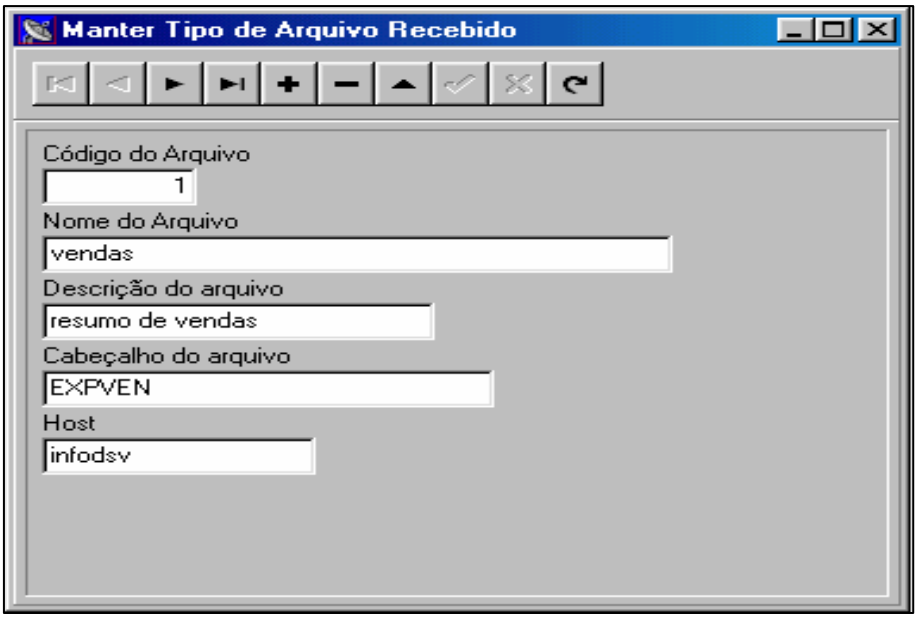
A último opção do menu (CADASTRO) é utilizado para fazer a manutenção dos Servidores que o protótipo utiliza, conforme figura 5.4.4. As interfaces das funções do menu (CADASTRO) são apresentadas a seguir.

FIGURA 5.4.4: MENU CADASTRO



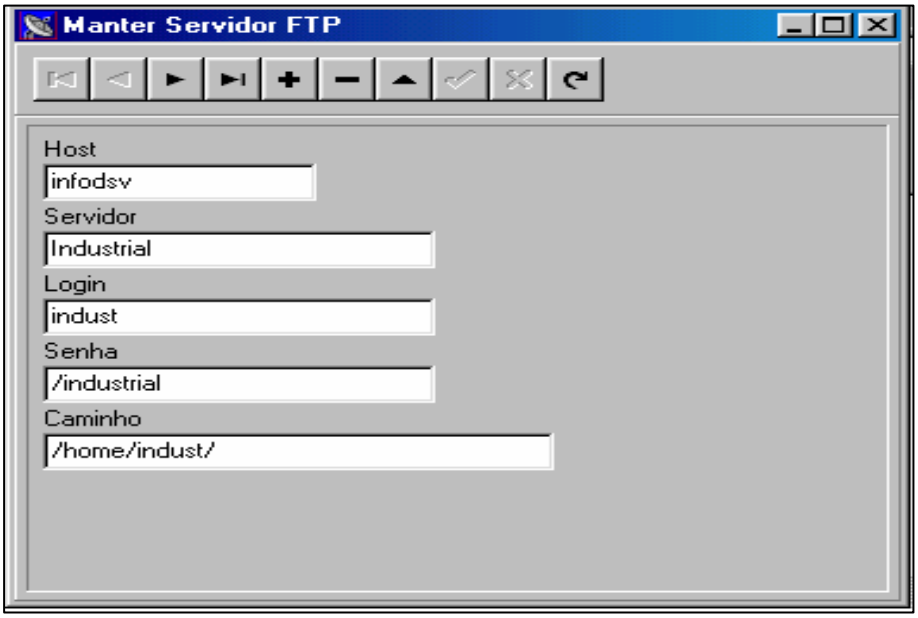
Conforme figura 5.4.5, apresenta a Interface de Arquivo Recebido, a qual serve para o usuário fazer manutenção dos tipos de arquivos que ele deseja receber e para onde deseja enviar o arquivo(Campo *Host*).

FIGURA 5.4.5 : INTERFACE PARA CADASTRO DE ARQUIVOS RECEBIDOS



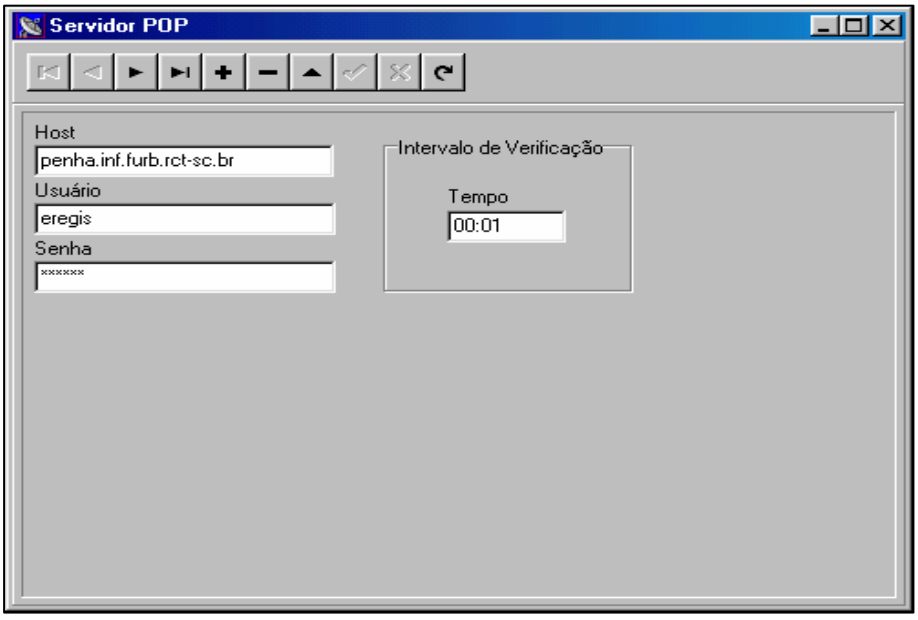
A interface Manter Servidor FTP serve para o usuário cadastrar os servidores FTP, que receberão os arquivos por e-mail conforme a figura 5.4.6

FIGURA 5.4.6 : CADASTRO DE SERVIDORES FTP



Para cadastrar o Servidor POP existe a interface Servidor POP conforme figura 5.4.7, onde é informado o endereço do Servidor para buscar os *e-mails*, o usuário e a senha para a acesso na caixa postal e o intervalo de tempo para verificar se existe mensagem na caixa postal.

FIGURA 5.4.7 - INTERFACE SERVIDOR POP



6 CONCLUSÃO

A conclusão deste trabalho foi conseqüência do comprimento de todas as etapas, uma de cada vez, seguindo-se rigorosamente o plano de desenvolvimento, conforme cronograma apresentado e aprovado na proposta de TCC.

O presente trabalho, permitiu a realização de estudo sobre a Arquitetura Internet, sua importância no mercado de trabalho, conceitos, abordagem histórica e características principais. Também permitiu um estudo mais aprofundado sobre os Protocolos de Correio Eletrônico, por estarem diretamente relacionados com os objetivos do trabalho, proporcionando os conhecimentos necessários para a elaboração do protótipo.

No levantamento bibliográfico foi constatado que, apesar de haver várias fontes de pesquisa sobre protocolos de Aplicações da Internet, poucas se aprofundaram sobre protocolo *Post Oficce Protocol* (POP). Houve carência de fontes bibliográficas que se definem a funcionalidade do protocolo POP.

O ambiente visual Delphi que utiliza a linguagem Pascal orientado a objeto, foi de fácil entendimento, não necessitando um estudo aprofundado sobre esta linguagem.

O uso de um gerenciador de banco de dados Paradox foi muito produtivo para a utilização neste trabalho. Não requer manipulações consideradas complexas, ou seja, com grande quantidade de informações e integrações ou vínculos entre tabelas. Desta maneira, servia apenas para o armazenamento dos dados, não exigindo muito das funções de manipulação dos dados armazenados, além de ser uma ferramenta acompanhada pelo ambiente Delphi na sua instalação.

O protótipo de software desenvolvido neste trabalho atingiu seus objetivos, permitindo então automatizar o processo de tratamento de mensagens recebidas pela Internet com arquivos anexados e distribuí-los para os Servidores destinatários.

A implementação do protótipo não se preocupou com o aspecto de segurança e a decodificação das mensagens, devido o fato destas questões estarem fora dos objetivos do trabalho, mas que poderiam se consideradas também.

6.1 EXTENSÕES

Para que este trabalho tenha continuidade, sugere-se conhecimento teórico da Arquitetura Internet e também conhecimento do funcionamento dos protocolos Post Office Protocol (POP), File Transfer Protocol (FTP) e Simple Mail Transfer Protocol.

Este trabalho poderá ser acrescentado de algum algoritmo de criptografia de dados para segurança no tráfego de informações e também estudo do RFC que trata a decodificação de mensagens, pois esse protótipo de software está preparado apenas para recepção de mensagens com arquivo anexados no formato texto.

REFERÊNCIAS BIBLIOGRÁFICAS

[BOR1995] BORLAND INTERNATIONAL INC. **Paradox para Windows – primeiros passos.** Scotts Valley : Borland, 1995.

[BRO1995] BROWNE, Steve. **Explorando a Internet com o mosaic através da World-Wide Web**. Rio de Janeiro : Infobook, 1995.

[CAN1996] CANTÚ, Marco. **Dominando o Delphi 2.0 "a bíblia".** São Paulo : Makron Books, 1996.

[CAR1994] CARVALHO, Tereza Cristina Melo de Brito. **Arquiteturas de redes de computadores OSI e TCP/IP.** São Paulo : Makron Books, 1996.

[COM1995] COMER, Douglas E. **Interligação em rede com TCP/IP.** Rio de Janeiro : Campus, 1995.

[HAH1995] HAHN, Hare**. Dominando a Internet.** São Paulo : Makron Books, 1995.

[FIS1990] FISHER, Alan S. **Case: utilização de ferramentas para desenvolvimento de software.** Rio de Janeiro : Campus, 1990.

[LEV1995] LEVINE, John R.BAROUDI, Carol. **Internet para leigos**: um manual para novos usuários. São Paulo: Berkeley, 1995.

[LUN1997] LUNA, André. **Internet fácil** Guia da Internet.br, mai.1997.

[MAR1996] MARTIN, James. **Analise e projeto orientados a objeto.** São Paulo : Makron Books, 1996.

[MEN1989] MENDES, SUELI. **Métodos para especificação de sistemas.** São Paulo : Editora Edgard Blucher, 1989.

[MIT1997] MIT, Laboratory for Computer Science 1997. www.pmg.lcs.mit.edu

[SAR1999] SARAIVA, Armando dos Santos. **Programando em Oracle**. Rio de Janeiro : I Infobook, 1999.

[STE1993] STEVENS, David L. COMER, DOUGLAS E**. Internet Working with TCP/IP volume III**: client-server programming and aplications BSD socket version. New Jersey: A Simon & Schuster Company EnglewwodCliffs, 1993.

[SWA1996] SWAN, Tom. **Delphi : Bíblia do programador.** São Paulo : Berkeley Brasil, 1996.

[TAN1996] TANENBAUM**,** Andrew S. **Redes de computadores.** Rio de Janeiro: Campus, 1996.