Análise de segurança em aplicações que utilizam plataformas UNIX e *MS-Windows* no modelo Cliente/Servidor

João Carlos Curti

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Análise de segurança em aplicações que utilizam plataformas UNIX e *MS-Windows* como Clientes e Servidores

João Carlos Curti

agosto de 2004

Banca Examinadora:

Prof. Dr. Paulo Lício de Geus (Orientador)
Instituto de Computação, UNICAMP

Profa. Dra. Maria de Fátima Ridolfi Pires O. da Silva
Coordenadoria Geral de Informática, UNICAMP

Prof. Dr. Ricardo de Oliveira Anido
Instituto de Computação, UNICAMP

Prof. Dr. Ricardo Dahab (Suplente)
Instituto de Computação, UNICAMP

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Curti, João Carlos

C94a Análise de segurança em aplicações que utilizam plataformas UNIX e MS-Windows como Clientes e Servidores / João Carlos Curti -- Campinas, [S.P. :s.n.], 2004.

Orientador : Paulo Lício de Geus

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Redes de computação – Medidas de segurança. 2. Cliente/Servidor (Computação). 3. TCP/IP (Protocolo de rede de computação). I. Geus, Paulo Lício. II. Universidade Estadual de Campinas. Instituto de Computação . III. Título.

Análise de segurança em aplicações que utilizam plataformas UNIX e *MS-Windows* como Clientes e Servidores

Este exemplar corresponde à redação final do Trabalho Final devidamente corrigida e defendida por João Carlos Curti e aprovada pela Banca Examinadora.

Campinas, 28 de setembro de 2004.

Prof. Dr. Paulo Lício de Geus
Instituto de Computação, UNICAMP
(Orientador)

Trabalho Final apresentado no Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Computação na Área de Redes de Computadores.

Agradecimentos

Aos meus pais, primeiramente, por terem me agraciado com o dom da vida, por sua paciência para transmitir seus ensinamentos, por sua luta diária e constante com extrema dedicação, pilares fundamentais desta conquista.

À minha namorada Luciana por sua compreensão, pelos dias distantes dedicados a este trabalho e o apoio nos momentos difíceis.

Ao meu Orientador, prof. Paulo, por sua atenção e aceitar este desafio, também por suas horas dedicadas a este trabalho.

Aos membros da banca examinadora por terem aceitado prontamente ao convite, pessoas as quais tenho muito respeito e admiração.

A todos amigos que auxiliaram direta ou indiretamente para a elaboração deste documento, oferecendo informações, amizades e experiências enriquecedoras.

A Universidade Estadual de Campinas que me acolheu e propiciou o crescimento pessoal de profissional.

Resumo

Nos últimos anos temos acompanhado e assistido as profundas mudanças na área de tecnologia, tanto em software quanto em hardware, permitindo que os computadores se popularizassem e que aplicações pudessem ser criadas para atender a crescente demanda deste novo mercado.

Infelizmente a adição dos mecanismos para prover segurança a estas aplicações não acompanhou o ritmo deste crescimento. Soluções foram adotadas para minimizar a exposição dos dados, uma vez que, com a utilização de computadores em rede, existe a necessidade do tráfego de informações por estes canais novos e de certo modo pouco explorados --- para não dizer pouco conhecidos --- para a maioria dos programadores.

Da diversidade de ambientes operacionais, o mercado e o marketing das empresas selecionaram duas plataformas distintas; uma proprietária e fechada, representada basicamente pela *Microsoft* e outra representada pelas variações de UNIX, sejam elas de código aberto (*FreeBSD, Linux, OpenBSD* e outros), ou proprietárias (*HP Tru64Unix, SunOs* e outros).

Sendo assim as informações tratadas pelas aplicações estão sujeitas, além de às suas próprias falhas, também às vulnerabilidades do Sistema Operacional onde são desenvolvidas e executadas.

Em vista destes fatos, o foco desta dissertação é apresentar como é possível a implantação de produtos de software comerciais do tipo ERP[1], independentemente da plataforma adotada pela empresa/instituição para execução de suas aplicações, provendo segurança nos canais de comunicação. Utiliza-se para isso ferramentas proprietárias ou nativas do próprio Sistema Operacional, bem como buscando de software livre para aplicações do tipo Cliente/Servidor e WEB.

[1] ERP – *Enterprise Resource Planning*

Abstract

In the last years we have followed and watched deep changes in technology, as much in software as in hardware, allowing for computers to get popular and for applications to be created to serve the growing demand of this new market.

Unfortunately the addition of mechanisms to supply security to applications not follow the pace of this growth; solutions were adopted to minimize data exposure, since, with the use of computer networks, there is a need for information flow through these new, somehow unexplored and even unknown channels for the majority of programmers.

From the diversity of operating environments, the market and the companies' marketing departments selected two distinct platforms: a proprietary and closed one, represented basically by *Microsoft* and another, represented by the variations of UNIX, be they of open code nature (*FreeBSD*, *Linux, OpenBSD etc*), or of proprietary nature (*HP Tru64Unix, SunOS etc*).

As such, the information dealt with by applications are subject not only to their own vulnerabilities, but also to those of the operating system for which they were developed.

In view of these facts, the focus of this dissertation is to show how it is possible, mainly for commercial implementation of ERP software and independently from the platform or operating system used, to supply security to the communication channels by using either free or native proprietary tools of the operating system itself, both to WEB as well as to other client/server applications.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Quando os primeiros computadores comerciais começaram a surgir, apenas grandes corporações e algumas instituições governamentais, possuíam ambiente e condições técnicas para a construção de aplicações, mesmo assim não era uma tarefa fácil, os equipamentos eram bastante limitados quanto ao uso de seus recursos e, as linguagens de programação existentes, permitiam que apenas os profissionais altamente capacitados fossem capazes de operar tais equipamentos.

Com o surgimento dos computadores pessoais, no final da década de 70, abriu-se uma nova perspectiva. Aplicações de pequeno porte poderiam ser geradas para executar nestes equipamentos, porém seria ainda necessário aguardar mais de uma década para que estes microcomputadores se tornassem populares, tanto nas empresas quanto em residências.

Até então, não havia preocupações significativas com relação ao tráfego de dados, a maior preocupação estava no armazenamento seguro das informações, uma vez que, nas poucas instituições dotadas de computadores ligados em rede, os clientes eram terminais conhecidos e a estrutura era montada por empresas que possuíam protocolos e ambientes proprietários.

A partir do final da década de 80 e início da década de 90 começam a se formar no mercado condições para o surgimento de novas aplicações, impulsionadas por três fatores principais:

1) A popularização dos microcomputadores nas empresas, no primeiro instante e posteriormente nas residências, através da redução dos custos de fabricação dos computadores e a adição de Sistemas Operacionais mais amigáveis. Fruto principalmente da parceria realizada entre a *Microsoft*®, que fornecia o MS-DOS e posteriormente as versões do *Windows* e a Intel®, que desenvolveu novas tecnologias de construção de chips e processadores, dando origem a família i386.

2) O surgimento de linguagens de programação mais amigáveis e voltadas para esta plataforma, facilitando o desenvolvimento de aplicações e permitindo que profissionais de informática, mesmo sem o conhecimento do conjunto de instruções, pudessem utilizar uma linguagem de alto nível.

3) O desenvolvimento das tecnologias de periféricos, armazenamento e rede que possibilitaram o surgimento de novas empresas para fornecimento de hardware para esta plataforma, sem a necessidade de tecnologia proprietária. No item de redes de computadores pode-se destacar a consolidação do protocolo TCP/IP como padrão de *facto* para construção de aplicações do tipo Cliente/Servidor e WEB. Uma das conseqüências foi a enorme expansão da Internet, que permitiu que computadores pudessem trocar dados utilizando qualquer tecnologia através de redes geograficamente distantes e heterogêneas.

Hoje temos um cenário diferente para o desenvolvimento e execução de aplicações. A segurança das informações que trafegam e ficam armazenadas nestes equipamentos é fator decisivo. Se por um lado não temos mais como deixar os computadores fora de uma rede, por outro deve-se possuir mecanismos que minimizem a exposição de informações, sigilosas ou mesmo valiosas, num ambiente vulnerável.

Até recentemente, a segurança era vista como mais uma das fases finais de elaboração de um produto de software, no entanto, os custos associados a falta de segurança são muito grandes: manutenções complexas e custosas, falhas de produtividade, fugas de informação, perda de contratos e gasto imprevisto de recursos de comunicação, entre outros. Além disso, podemos mencionar outros fatores que agravam a situação como:

Complexidade e integração de programas – as aplicações não executam de forma solitária: agora fazem parte de um ambiente complexo com inúmeras interações entre aplicações. Hoje o código necessário para produzir uma aplicação sofisticada e multifuncional é gigantesco.

Necessidade de produzir software rapidamente – A necessidade de lançar novas versões de software, ditada por razões de marketing e financeiras, contribui também significativamente para este problema. Por essa razão, muitos produtos disponíveis atualmente estão repletos de códigos que apenas cumprem parcialmente a sua função, mas não são soluções conceitualmente elegantes, a prioridade

reside em colocar para funcionar rapidamente, não importando muito os meios usados. Além disto, os prazos curtos não condizem com fases de teste e revisão do produto.

Correções de código de fonte fechada – O cliente final de uma aplicação de código de fonte fechada é obrigado a aceitar as correções produzidas pelo fabricante e não dispõe de alternativas. O cliente, em geral, não possui condições para confirmar se as correções divulgadas realmente resolvem o problema ou se geram novos problemas.

Preparação inadequada dos programadores – Muitos fabricantes não compreendem os riscos a que expõem os seus clientes quando desenvolvem produtos deficientes. Na raiz deste subproblema parecem estar a preparação inadequada dos programadores, em parte devido ao uso excessivo de ciclos "produção – depuração – revisão" na codificação de software em contra partida a práticas orientadas ao *design*, estilo de programação, auditorias de código entre outros.

De fato, houve uma grande evolução com a criação de mecanismos de modo a prover ao protocolo TCP/IP meios para adicionar segurança no canal por onde os dados trafegam, também observa-se o esforço das organizações de software livre em oferecer alternativas viáveis para suprir as deficiências ainda existentes para execução segura de aplicações em plataformas proprietárias.

No decorrer deste trabalho serão apresentadas, através da análise de segurança, quais são os problemas encontrados pelas aplicações para se comunicar com segurança em uma rede de computadores, composta por elementos mantidos por soluções proprietárias e de software livre.

1.1 Apresentação do problema

Para entender como as aplicações são construídas e como trafegam os dados numa rede do tipo TCP/IP – protocolo principal utilizado nas redes atuais, é necessário conhecer a sua estrutura, compreender como estas informações chegam a camada de aplicação e são tratadas.

Pode-se iniciar a exposição deste assunto utilizando como exemplo uma aplicação simples, que não requer nenhuma outra implementação adicional –apenas para ilustrar o processo, em contraste com aplicações mais complexas, que necessitam de pesquisas e conhecimentos mais amplos, não somente nesta arquitetura, mas também em tecnologias e protocolos proprietárias de redes.

Na Figura 1.1 conforme [40] há um exemplo da estrutura em camadas da arquitetura TCP/IP e também das interfaces entre estas camadas. Utiliza-se neste exemplo como meio físico a tecnologia

Ethernet, principal tecnologia de rede local utilizada nas empresas/instituições, bastante conhecida e instalada e a aplicação FTP[1], que funciona de modo Cliente/Servidor.

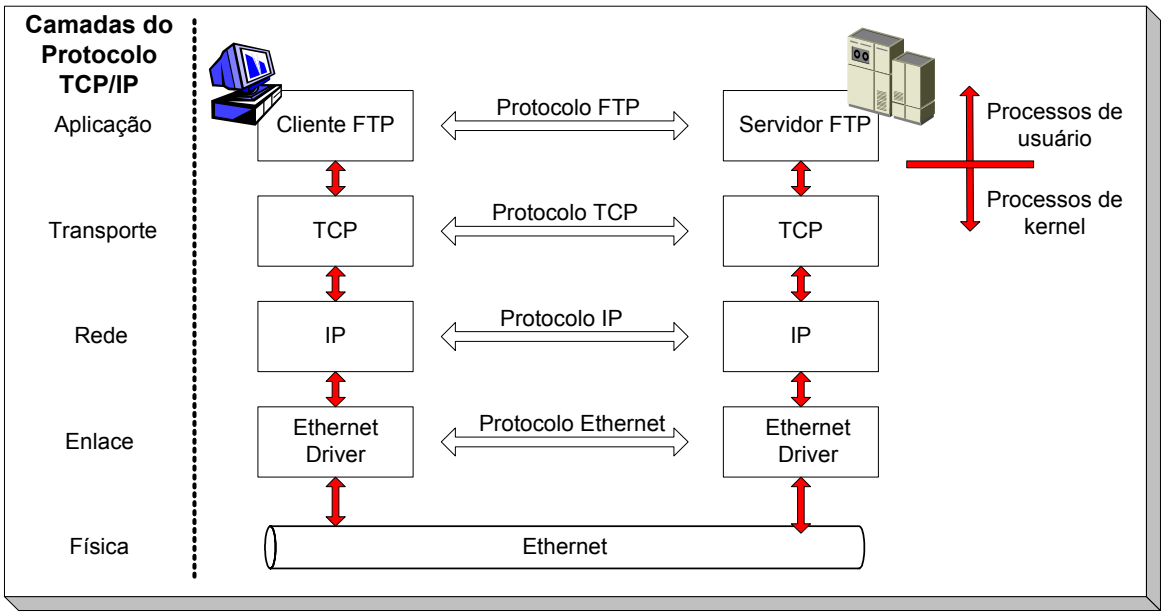

Figura 1.1 – Arquitetura TCP/IP

Basicamente, para se ter uma troca de informações em rede é necessário no mínimo uma estrutura como a apresentada, ou seja, duas máquinas ligadas por uma tecnologia que utilizam o mesmo protocolo de comunicação. Em muitos casos, a Internet é o meio utilizado para interligar os computadores de redes locais geograficamente distantes. A pilha de protocolos Internet é constituída por camadas lógicas nas quais se definem protocolos de comunicação.

A maneira utilizada para fazer chegar a informação entre a aplicação de origem através da rede até a aplicação destino, baseia-se no encapsulamento sucessivo dos dados em pacotes de informação, em função dos protocolos utilizados na origem. Para tal, constrói-se um novo pacote adicionando um cabeçalho adequado ao pacote do protocolo anterior. No destino, executa-se um processo inverso de desencapsulamento dos pacotes de informação (remoção de cabeçalhos) até se restituir aos dados a sua forma original, a qual será tratada pela aplicação receptora.

A Figura 1.2 [40] apresenta este processo numa rede baseada em tecnologia *Ethernet*. Durante a travessia na rede, os dados circulam em pacotes cujo formato depende dos meios físicos usados, mas contêm todo o encapsulamento executado na origem, os dispositivos de interligação de redes podem

[1] FTP: *File Transfer Protocol*, RFC nº 959

assim observar os pacotes em trânsito e tomar decisões de encaminhamento baseando-se nos cabeçalhos desses pacotes.

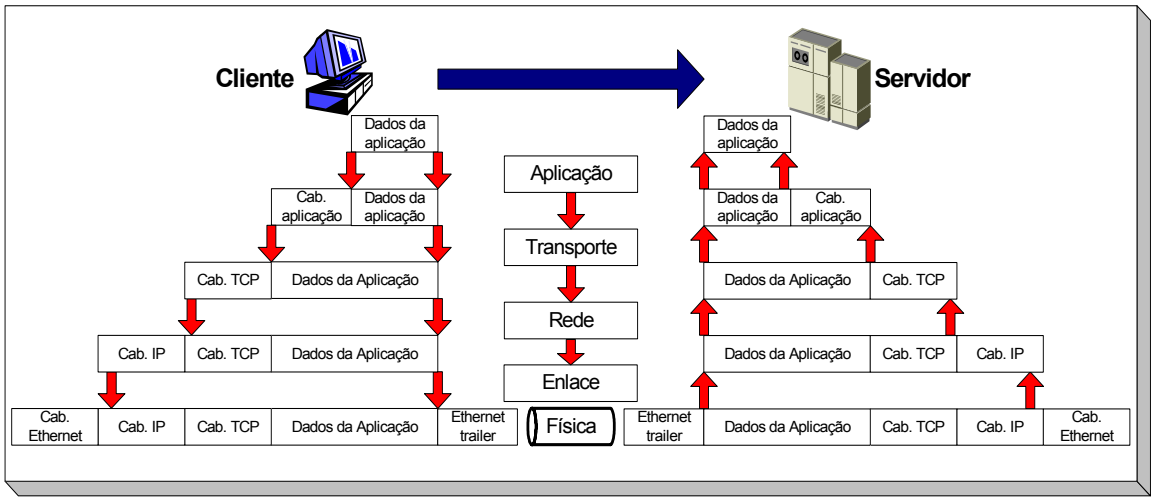

Figura 1.2 – Encapsulamento TCP/IP de pacotes do Cliente para um Servidor

O processo para comunicação pode ser simples, uma máquina "A" abrindo uma conexão com uma máquina "B", utilizando uma tecnologia para redes locais, porém na realidade existe uma infinidade de tecnologias de redes e protocolos de comunicação.

Deste modo é necessário ampliar esta estrutura de forma a expor a complexidade encontrada fora dos domínios de uma rede local, onde existem outras redes, conhecidas e desconhecidas, com filtros (*firewalls*), roteadores e outros elementos que estarão no caminho entre as máquinas "A" e "B".

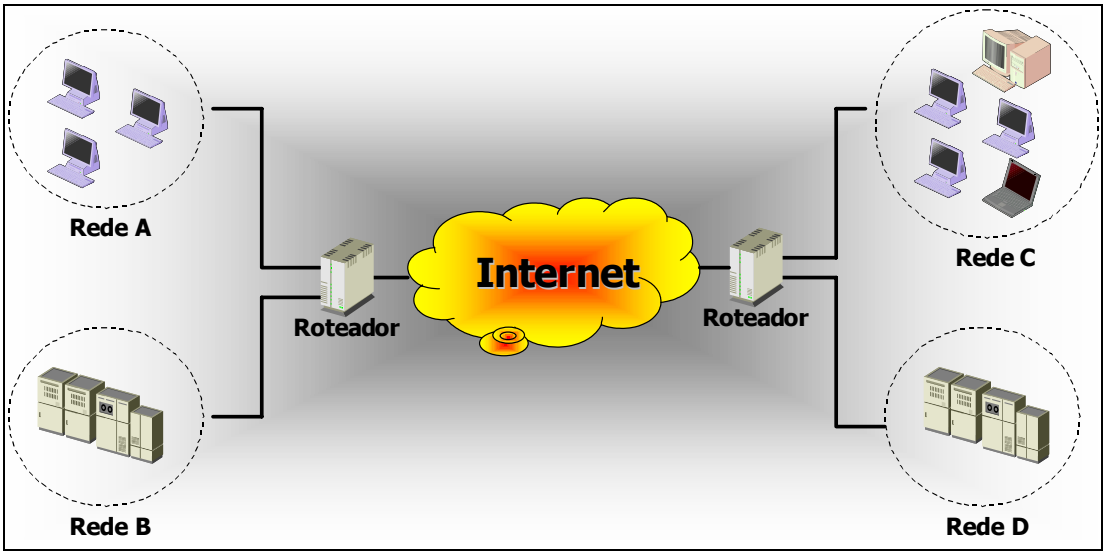

Figura 1.3 – Integração de redes via Internet

Internamente numa rede local, existem meios para se mensurar os riscos de exposição dos dados, pelo fato de termos o acesso ao controle e a administração dos elementos que podem ser considerados como pontos vulneráveis. Mas quando estas informações precisam sair deste domínio controlado e trafegar por redes desconhecidas, não há como garantir a confiabilidade no caminho que os dados percorrem da origem ao seu destino.

A proposta desta dissertação é mostrar que aplicações cliente/servidor desenvolvidas, principalmente com finalidades comerciais, necessitam de cuidados especiais na área de segurança. Além disso, deve-se considerar que aplicações complexas requerem, na maioria dos casos, soluções complexas, obtidas através de um estudo amplo das opções de soluções disponíveis na plataforma desejada a fim de alcançar os melhores resultados.

Sendo assim, este trabalho pretende apresentar um estudo considerando o ambiente de execução das aplicações e as vulnerabilidades as quais estão expostas, destacadas no Capítulo 5. O foco principal está na exposição das soluções viáveis para implementação de uma aplicação no modelo cliente/servidor, com características semelhantes as encontradas em nosso estudo, de forma segura através da rede de computadores, fazendo-se cumprir também os princípios que regem a comunicação entre computadores, além de possuir mecanismos que possibilitem que os elementos envolvidos na comunicação possam ser identificados – através de uma autenticação segura e irrevogável e, integrado por um sistema que se utiliza de métodos criptográficos para garantir a segurança durante a troca de informações através da rede.

1.2 Organização do trabalho

Durante a elaboração deste trabalho diversos aspectos da arquitetura cliente/servidor foram estudados e pesquisados. Por se tratar de um assunto abrangente iniciou-se já no Capítulo 1 com a apresentação do problema da comunicação entre aplicações que utilizam a arquitetura cliente/servidor no cenário mais comum encontrado atualmente, a rede TCP/IP.

No Capítulo 2 será abordado o modelo Cliente/Servidor, seus conceitos, arquiteturas, elementos e processos que descrevem esta tecnologia.

No Capítulo 3 é apresentado o cenário utilizado para comunicação entre as partes Cliente e Servidor das aplicações, com uma análise e descrição dos protocolos utilizados nas redes, aspectos envolvidos na conexão entre o cliente e o servidor, princípios que regem a comunicação. A questão da

implementação de segurança na comunicação é analisada e tratada no Capítulo 4. Neste Capítulo serão apresentados os protocolos seguros existentes e também protocolos de autenticação.

No Capítulo 5 faz-se a apresentação de um estudo de caso de uma aplicação comercial desenvolvida com ferramentas e linguagem para plataforma *MS-Windows*. Neste Capítulo são apresentadas também as tecnologias e soluções disponíveis de segurança para a arquitetura Cliente/Servidor, destacando-se as técnicas para criação de VPN[1]'s, Relacionamento de Confiança entre *hosts* ou domínios em redes com tecnologia *MS-Windows* e Certificação Digital, visando a solução do problema citado em nosso estudo de caso.

No Capítulo 6 é apresentada a análise da implementação de cada tecnologia, utilizando os conceitos vistos nos Capítulos 3, 4 e 5. O objetivo deste Capítulo é apresentar alternativas para solucionar os problemas relativos a segurança da informação, vulnerabilidade nos participantes da comunicação e autenticidade na aplicação Cliente/Servidor mostrada no estudo de caso, uma vez que a aplicação em estudo nativamente não possui mecanismos para prover segurança para execução em um cenário complexo, inclusive para a versão para a WEB desta mesma aplicação.

Por fim, no Capítulo 7 são apresentadas algumas considerações e conclusões gerais obtidas ao longo de todas as etapas abordadas em cada Capítulo e também abre-se um novo cenário para pesquisa utilizando a tecnologia de VPN para solução do estudo de caso.

[1] VPN : Virtual Private Network

Capítulo 2

O Modelo Cliente/Servidor

Neste capítulo serão expostos os conceitos e elementos de um modelo Cliente/Servidor, suas principais características e as arquiteturas de interação entre os participantes da comunicação. A escolha do modelo e o modo como são realizados os processamentos, influenciam diretamente no desempenho das aplicações.

Atualmente existe uma grande flexibilidade e diversidade destes modelos e muitos são utilizados nas aplicações que executamos diariamente. A exposição destes modelos e arquiteturas nos faz compreender melhor o papel desempenhado em cada uma das extremidades da comunicação entre computadores.

2.1 Elementos do modelo Cliente/Servidor

A maior parte das aplicações de rede é desenvolvida assumindo-se que um dos agentes comunicantes é o cliente e o outro é o servidor, sendo o objetivo da aplicação fornecer um serviço pré-definido através do servidor. Neste sentido a arquitetura Cliente/Servidor vem sendo desenvolvida há vários anos, porém em pequenos passos.

Primeiro, a realocação de aplicações em *Mainframe* para as chamadas plataformas abertas rodando Sistema Operacional UNIX. Posteriormente, com relação a abordagem dos dados, saindo de Sistemas de Arquivos ou Banco de Dados Hierárquicos locados em *Mainframes* para Sistemas de Banco de Dados Relacional e posteriormente, a importância da capacidade gráfica dos pacotes de "*front-end*" existentes, facilitando a interação com o usuário [20].

Através dos estudos realizados do modelo Cliente/Servidor, pode-se destacar como mais comuns as seguintes definições:

- O termo Cliente/Servidor refere-se ao método de distribuição de aplicações computacionais através de muitas plataformas., tipicamente essas aplicações estão divididas entre um provedor de acesso, uma central de dados e numerosos clientes contendo uma interface gráfica para usuários para acessar e manipular dados.
- Cliente/Servidor geralmente refere-se a um modelo onde dois ou mais computadores interagem de modo que um oferece serviços aos outros. Este modelo permite aos usuários acessarem informações e serviços de qualquer lugar.
- Cliente/Servidor é uma arquitetura computacional que envolve requisições de serviços de clientes para servidores. Uma rede Cliente/Servidor é uma extensão lógica da programação modular.
- Uma definição para a arquitetura Cliente/Servidor seria a existência de uma plataforma base para que as aplicações, onde um ou mais Clientes e um ou mais Servidores, juntamente com o Sistema Operacional e o Sistema Operacional de Rede, executem um processamento distribuído.

Em suma, um sistema Cliente/Servidor poderia ser, então, entendido como a interação entre Software e Hardware em diferentes níveis, implicando na composição de diferentes computadores e aplicações de forma distribuída. Para melhor se entender o paradigma Cliente/Servidor é necessário observar que o conceito chave está na ligação lógica e não física. O Cliente e o Servidor podem coexistir ou não na mesma máquina [28].

Um ponto importante para uma real abordagem Cliente/Servidor é a necessidade de que a arquitetura definida represente uma computação distribuída [20]. Algumas das características do Cliente e do Servidor são descritas a seguir baseados em [31] e [12].

2.1.1 Cliente

Cliente, também denominado de "*front-end*" e "*Workstation*", é um processo que interage com o usuário através de uma interface gráfica ou não, permitindo consultas ou comandos para recuperação de

dados e análise e representando o meio pela qual os resultados são apresentados. Além disso, apresenta algumas características distintas:

- É o processo ativo na relação Cliente/Servidor.
- Inicia e termina as conversações com os Servidores, solicitando serviços distribuídos.
- Não se comunica com outros Clientes.
- Torna a rede transparente ao usuário.

2.1.2 Servidor

Servidor, também denominado "*back-end*", fornece um determinado serviço que fica disponível para todo Cliente que o necessita. A natureza e escopo do serviço são definidos pelo objetivo da aplicação Cliente/Servidor.

Além disso, ele apresenta ainda algumas propriedades distintas:

- É o processo reativo na relação Cliente/Servidor.
- Possui execução contínua.
- Recebe e responde às solicitações dos Clientes.
- Não se comunica com outros Servidores enquanto estiver fazendo o papel de Servidor.
- Presta serviços distribuídos.
- Atende a diversos Clientes simultaneamente.

Alguns tipos de serviços que um Servidor pode proporcionar são:

- Servidor de Arquivos
- Servidor de Impressão
- Servidor de Banco de Dados
- Servidor de Redes
- Servidor de Fax
- Servidor *X-Windows*
- Servidor de Processamento e Imagens
- Servidor de Comunicação e outros.

O estilo de interação entre o usuário e o Cliente não precisa, necessariamente, ser feita por poderosas interfaces gráficas. Porém, já que o poder de processamento local do Cliente está disponível,

pode-se retirar todo seu proveito, através de interfaces gráficas – GUI (*Graphical User Interface*), para melhor rendimento do usuário no seu trabalho.

2.2 Modelos da arquitetura Cliente/Servidor

Existem cinco tipos de modelos para a implantação da arquitetura Cliente/Servidor em processamentos distribuídos conforme [31]:

A primeira abordagem para um sistema distribuído é a arquitetura Cliente/Servidor Simples. Nesta arquitetura, o Servidor não pode iniciar nada, este somente executa as requisições do Cliente. Existe uma clara função de diferenciação, pode-se estabelecer que o Cliente é o mestre e o Servidor é o escravo, como mostra a Figura 2.1.

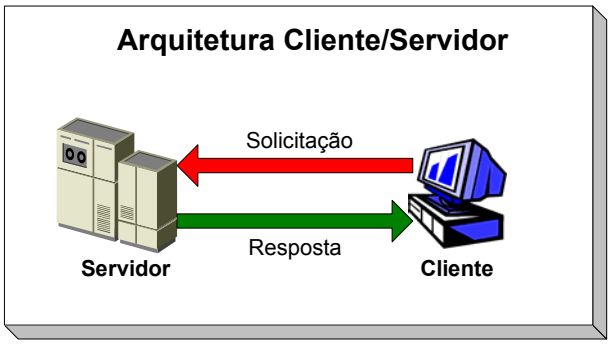

Figura 2.1 – Arquitetura C/S simples

2.2.1 Arquitetura C/S em Dois Níveis

A configuração usual Cliente/Servidor encontrada na maioria das empresas. É aquela em que existem vários Clientes requisitando serviços a um único Servidor. Esta arquitetura se caracteriza como sendo centrada no Servidor (Figura 2.2), porém na visão do usuário, ele imagina que existem vários servidores conectados a somente um cliente, ou seja, centrado no Cliente (Figura 2.3). Entretanto, com as várias ligações de comunicação possíveis, existe na realidade uma mistura de Clientes e Servidores (Figura 2.4).

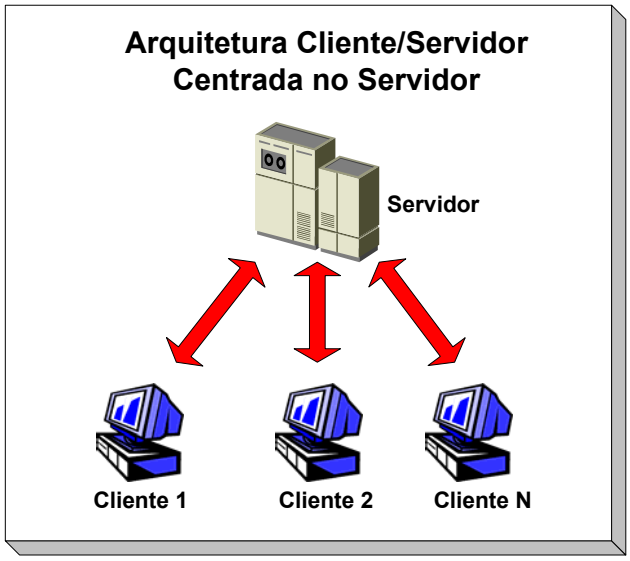

Figura 2.2 – Arquitetura C/S em dois níveis – Centrada no Servidor

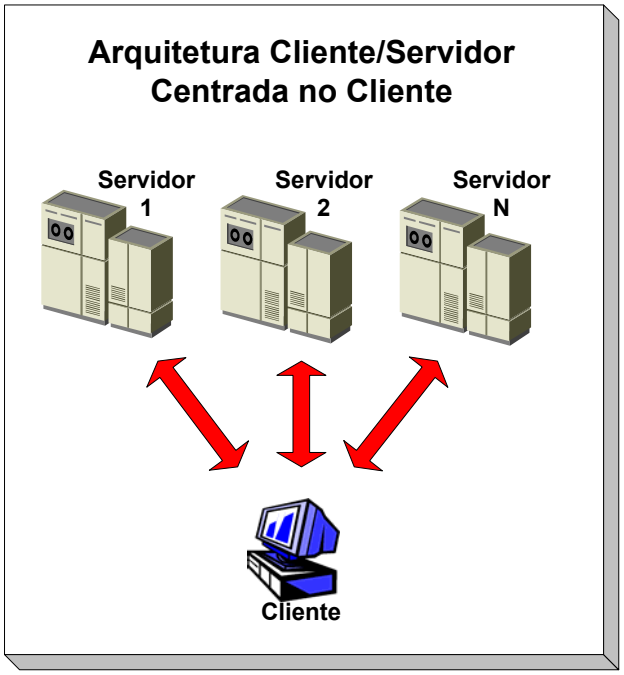

Figura 2.3 – Arquitetura C/S em dois níveis – Centrada no Cliente

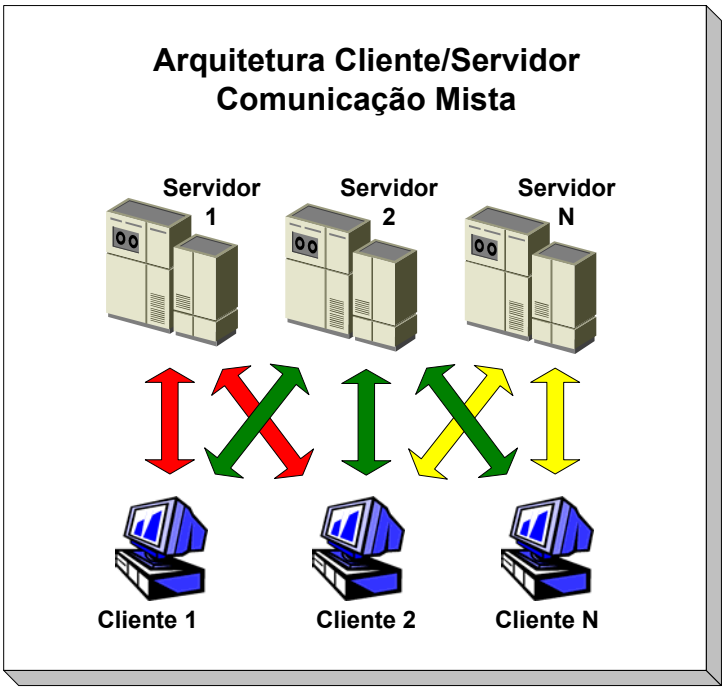

Figura 2.4 – Arquitetura C/S em dois níveis – Comunicação Mista

2.2.2 Arquitetura C/S Multinível

Nesta arquitetura, ilustrada na Figura 2.5, permite-se que uma aplicação possa assumir tanto o perfil do Cliente como o do Servidor, em vários graus, em outras palavras, uma aplicação em alguma plataforma será um Servidor para alguns Clientes e, concorrentemente, um Cliente para alguns Servidores.

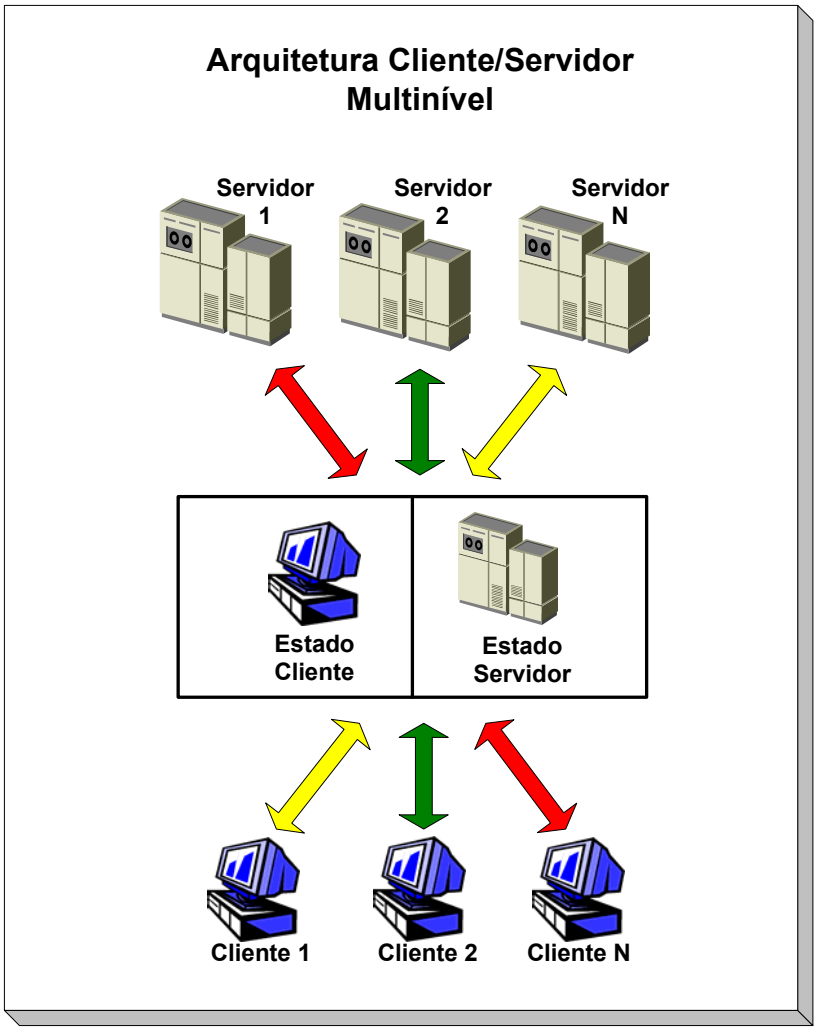

Figura 2.5 – Arquitetura C/S multinível

2.2.3 Arquitetura C/S Par a Par

Esta arquitetura pode ser vista como o caso mais geral da arquitetura Cliente/Servidor, ilustrado na Figura 2.6, cada um dos nodos desta arquitetura assume tanto o papel de Cliente quanto de Servidor. É o caso onde o processo interage com outros processos em uma base pareada, não existindo nenhum Mestre ou Escravo, qualquer estação de trabalho pode iniciar um processamento, caso possua uma interface de comunicação entre o usuário e o processo Cliente.

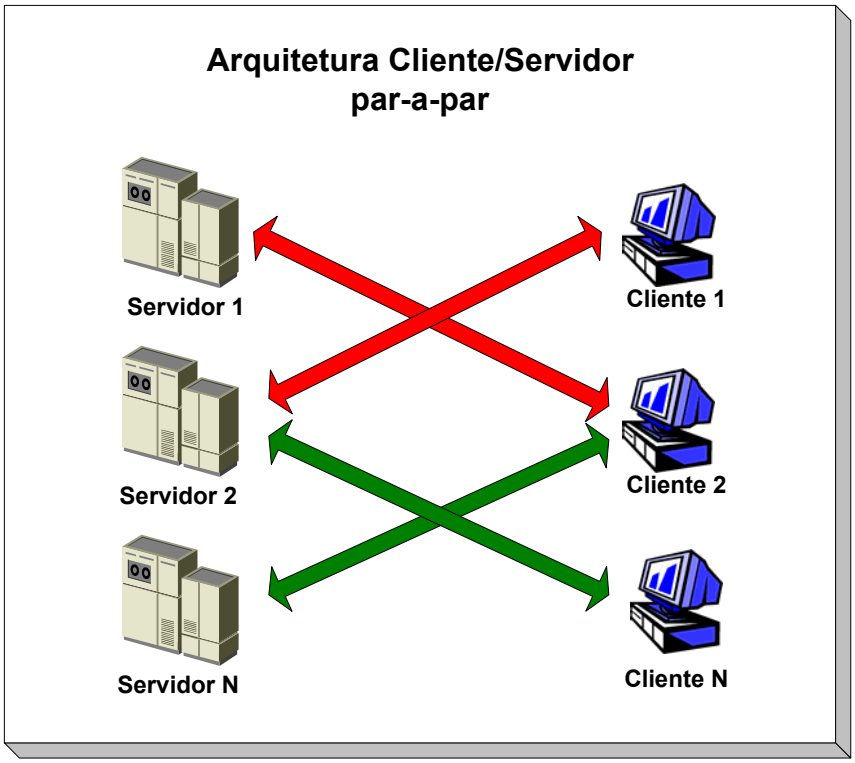

Figura 2.6 – Arquitetura C/S – Par a Par

2.3 O Processo distribuído da Arquitetura Cliente / Servidor

A arquitetura Cliente/Servidor divide uma aplicação em processos que são executados em diferentes máquinas conectadas a uma Rede de Computadores, formando um único sistema. O paradigma da tecnologia Cliente/Servidor serve como um modelo, entre outros, para interação entre processos de software em execução concorrente [02].

Os processos ou tarefas, a serem executadas são divididos entre o Servidor e o Cliente, dependendo da aplicação envolvida e das restrições impostas pelo Sistema Operacional de Rede (SOR), quanto mais avançado for o Sistema Operacional de Rede, menor será a aplicação em si, uma vez que a implementação do código para acessar a rede já se encontra definido no SOR.

No contexto do presente trabalho pretende-se utilizar as características do processamento distribuído. Este tipo de processamento apresenta duas configurações para uma arquitetura Cliente/Servidor. A primeira, que é representada por três camadas, é responsável pela visualização da interação entre os aplicativos e o hardware, como pode ser visto na Figura 2.7, já a segunda

representação, também fornecida em três camadas, mostra como é tratada a divisão da funcionalidade de uma aplicação, segundo as configurações do *Gartner Group*[1], como pode ser visto na Figura 2.8.

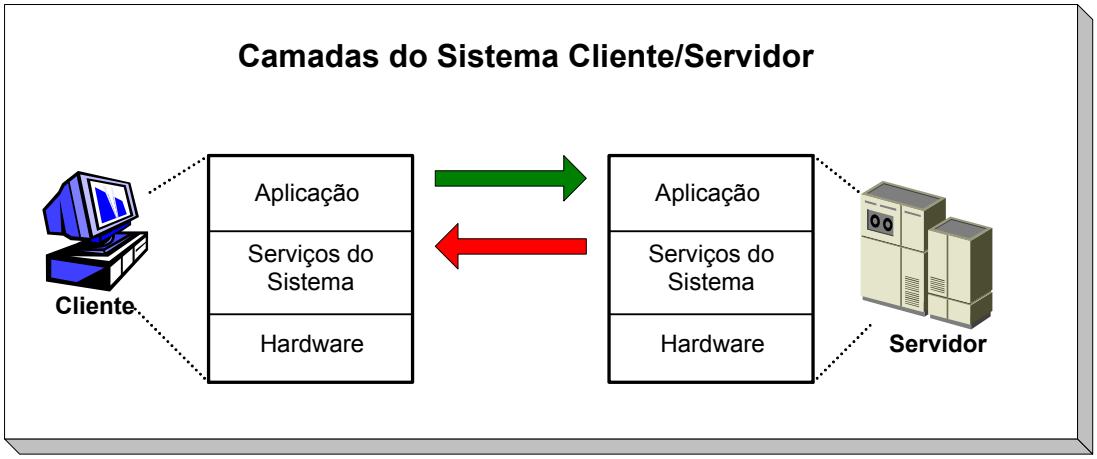

Figura 2.7 – Sistema Cliente/Servidor

2.3.1 Processamento Distribuído ou Concorrente

A distribuição de aplicações e tarefas se faz através de múltiplas plataformas de processamento. O processamento distribuído implica que essas aplicações/tarefas irão ocorrer em mais de um processo, na ordem de uma transação a ser concluída, em outras palavras, o processamento é distribuído através de duas ou mais máquinas e os processos, na maioria, não rodam ao mesmo tempo, por exemplo, cada processador realiza parte de uma aplicação em uma seqüência. Geralmente, o dado usado em um ambiente de processamento distribuído também é distribuído através de plataformas.

O processamento distribuído, também denominado de processamento concorrente utiliza-se do mecanismo de passagem de mensagens para a comunicação entre processos, que podem ser de três tipos básicos; Filtro, *Peer* (não hierárquico) e Cliente e Servidor [28], como mostrado na Figura 2.9.

[1] *Gartner Group*: www.gartnergroup.com

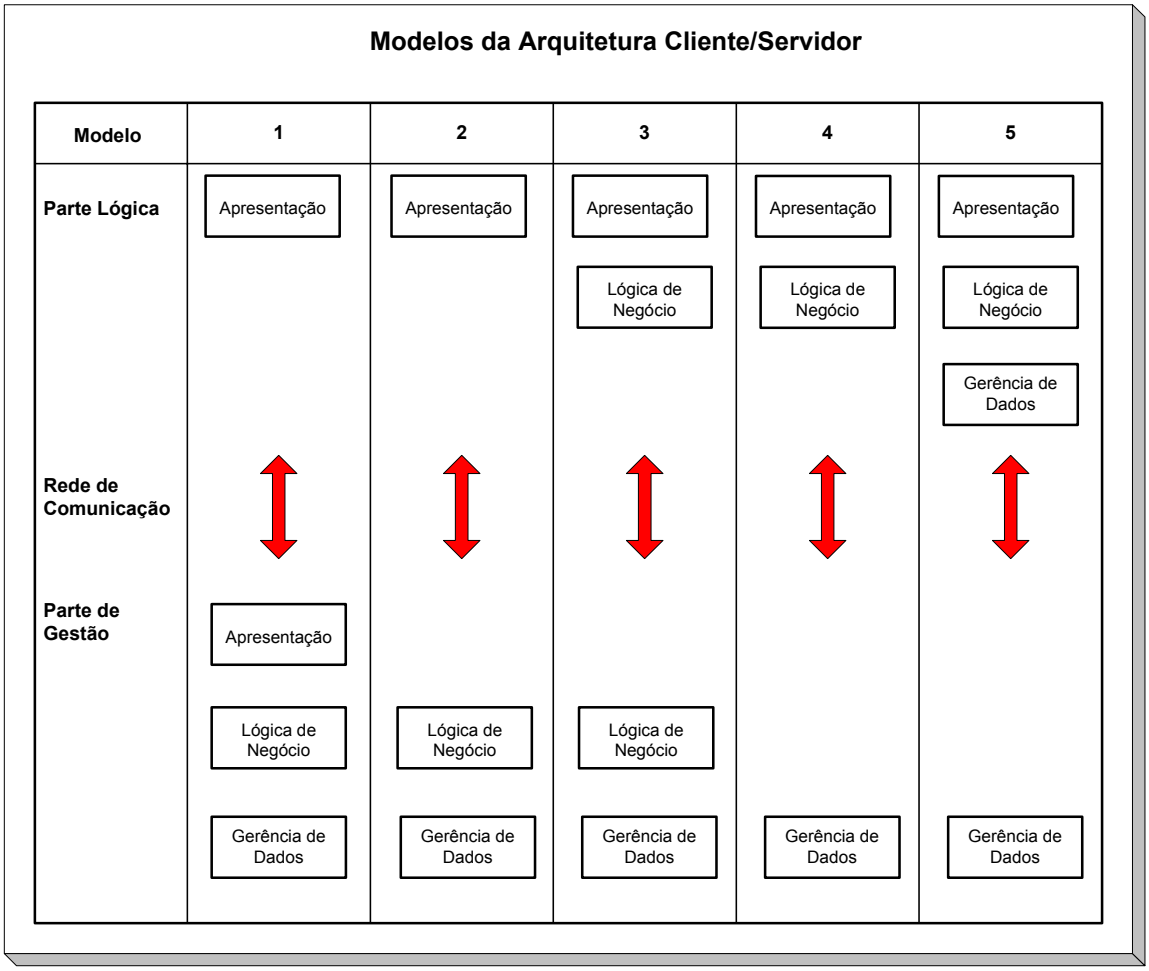

Figura 2.8 – Modelo de Distribuição de Processos

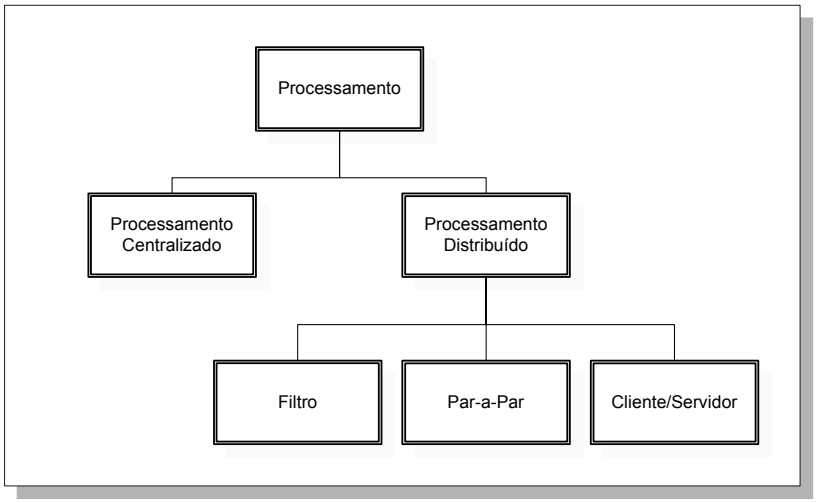

Figura 2.9 – Processamento Distribuído

Características do **Processo de Filtro**:

- Determina uma conversão na mensagem de comunicação entre o usuário e o *host*. Exemplo: Ligação de um *desktop* com um *mainframe* através de um emulador de terminal.

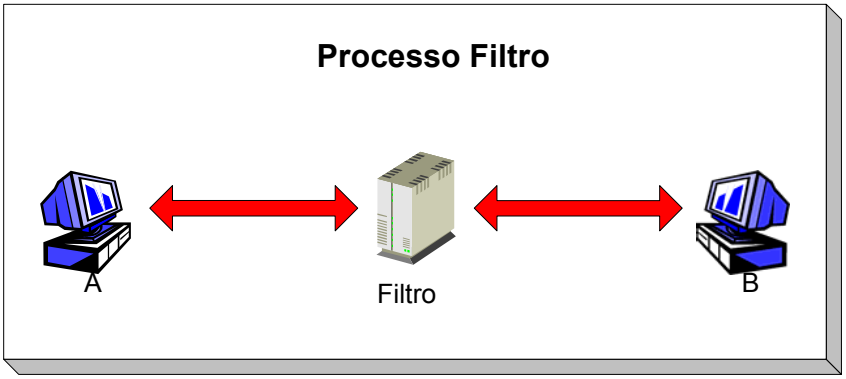

Figura 2.10 – Processo de Filtro

Características do **Processo *Peer-to-Peer*** (não hierárquico):

- São processos "clones" rodando em todas as máquinas e prestando serviços uns para os outros.
- Não existem processos servidores, estabelecendo um Servidor Dedicado.
- Cada processo pode ser Cliente e Servidor para outros processos.

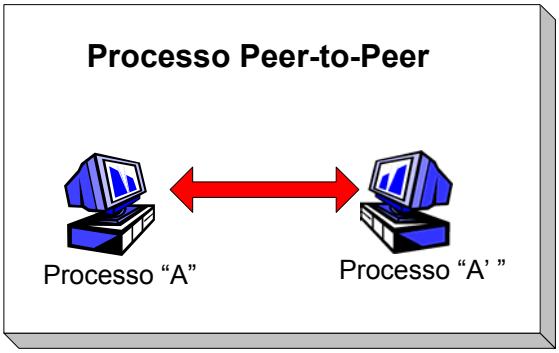

Figura 2.11 – Processo *Peer-to-Peer*

Durante a execução das consultas, os processos Cliente e Servidor são confundidos com os processos *Peer-to-Peer*, porque este processo foi desenvolvido com base na LU6.2 (*Logical Unit* versão 6.2) do SNA (*Systems Network Architecture*) da IBM [28].

Características do **Processo Cliente / Servidor**:

- Existem processos distintos: o processo cliente é diferente do processo servidor.
- Os processos servidores tornam a estação Servidora dedicada ao seu trabalho.

- Processos clientes são sempre clientes.
- Processos servidores podem se tornar processos clientes de outros Servidores.

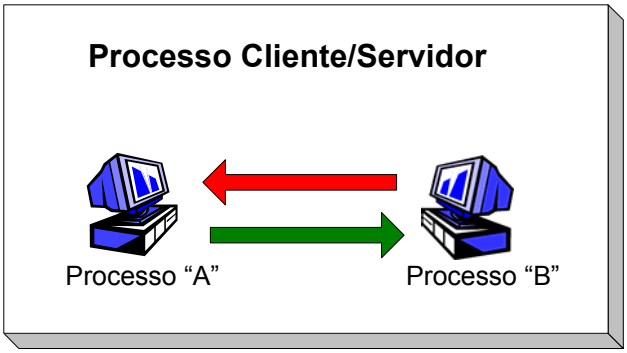

Figura 2.12 – Processo Cliente/Servidor

O modelo que utiliza processos Cliente/Servidor depende do cliente para inicializar a comunicação. Caso o Servidor comece a comunicação, o processo instalado fica sendo o *Peer-to-Peer*.

A característica básica da arquitetura Cliente/Servidor é a que processos Clientes enviam pedidos a um processo Servidor, que retorna o resultado para o Cliente. O processo Cliente fica então liberado da ação do processamento da transação podendo realizar outros trabalhos.

2.4 Camadas da Arquitetura Cliente/Servidor

A arquitetura Cliente/Servidor é dividida em três camadas básicas, como ilustra a Figura 2.13. A camada de Aplicação consiste dos processos da aplicação, entre eles, os processos Cliente e Servidor, a camada de Serviços de Sistemas compreende o Sistema Operacional (SO) e o Sistema Operacional de Rede (SOR), destinando-se ao controle do hardware, por último a camada de hardware, onde estão localizados os periféricos ligados aos Clientes e Servidores.

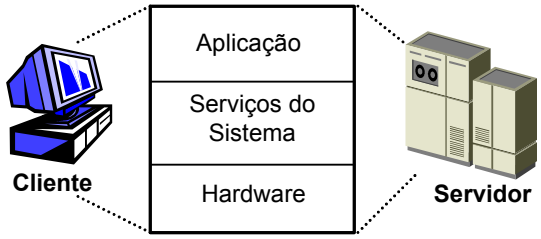

Figura 2.13 – Camadas da arquitetura C/S

A arquitetura Cliente/Servidor pode existir tanto no nível da camada de Aplicação, quanto no da camada de Serviços do Sistema. A coexistência do paradigma nestas camadas surge em função da hierarquia das atuações no sistema. Caso o "usuário" seja externo ao sistema, então os processos Cliente e Servidor compõem a camada da Aplicação, enquanto que, se o "usuário" for um programa de aplicação, o Cliente é um processo redirecionador e o Servidor será um processo respondedor da camada de Serviços do Sistema.

A utilização de sistemas Cliente/Servidor pela camada de aplicação utiliza Serviços do Sistema não Cliente/Servidor (Figura 2.13), entretanto, sistemas não Cliente/Servidor, ao nível da aplicação utilizam Serviços do Sistema Cliente/Servidor [28].

Para sistemas Cliente/Servidor na camada de aplicação, a camada Serviços do Sistema oferece somente um mecanismo de IPC[1] para troca de mensagens. Por outro lado, a camada Serviços do Sistema configurada como Cliente/Servidor, é responsável por gerenciar o redirecionamento das solicitações de gravação/leitura, por exemplo.

É importante notar que a diferença entre os sistemas Cliente/Servidor nas camadas de Aplicação e Serviços do Sistema, é o equilíbrio entre a quantidade de processamento tanto no lado do Cliente quanto no lado Servidor. Existem vários sistemas que podem ser baseados na estrutura Cliente/Servidor. O uso mais freqüente são as aplicações de Banco de Dados usando processos SQL (*Structured Query Language*) de *front-end*, para acessar remotamente, as bases de dados.

A Figura 2.14 mostra uma estrutura baseada num Servidor de Arquivos. Esta estrutura ocasiona um maior fluxo de informações na rede, uma vez que todo o arquivo será transferido para o Cliente para então ser trabalhado.

[1] IPC: *InterProcess Comunication*

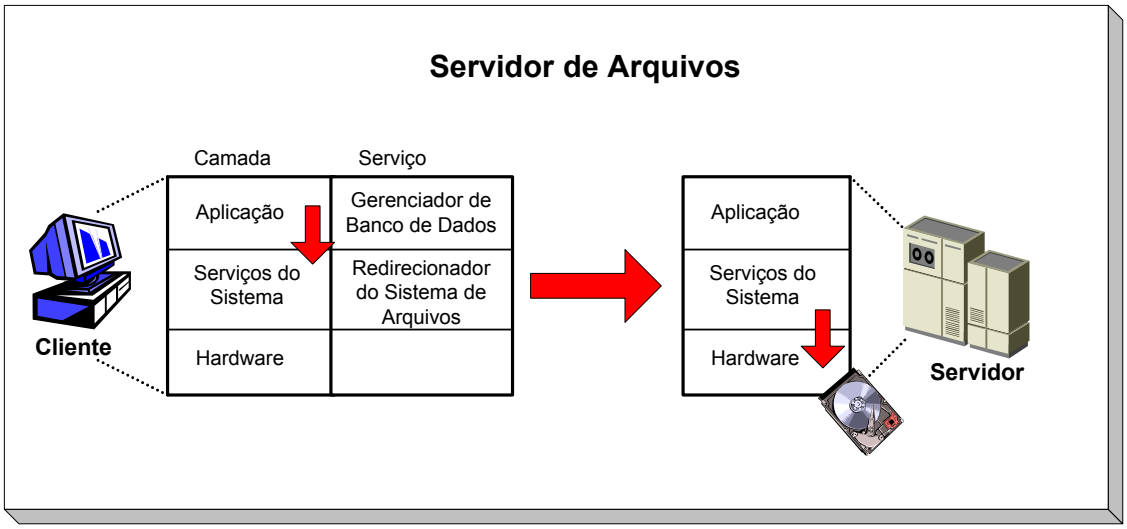

Figura 2.14 – Arquitetura C/S como Servidor de Arquivos

Neste tipo de estrutura, a camada de Aplicação passa a ser o Cliente do Sistema. Com isto, a camada de Serviço do Sistema é utilizada simplesmente como um redirecionador para acesso a base de dados [28]. Pode-se chamar este Sistema como falso Sistema Cliente/Servidor, por não haver um equilíbrio de processamento entre os dois lados Cliente e Servidor. O lado Servidor somente terá o trabalho de executar as rotinas comuns de I/O[1], não caracterizando assim, como um processamento intrínseco à aplicação.

A Figura 2.15 demonstra outra possibilidade de se estruturar um Sistema baseado na arquitetura Cliente/Servidor, que consiste na utilização de um Servidor de Banco de Dados dedicado, podendo coexistir normalmente com o Servidor de arquivos. Neste momento, com um Servidor de Banco de Dados exclusivo, o fluxo de informações trafegadas na rede diminui, já que, somente a resposta da consulta é retornada ao Cliente, ao invés de transferir todo o arquivo como era feito anteriormente.

O Cliente, neste tipo de estrutura é o usuário, passando a ter uma visão da aplicação como se as partes, Cliente e Servidor, fossem algo único. A Figura 2.16 exemplifica esta visão.

[1] I/O: *Input/Output*

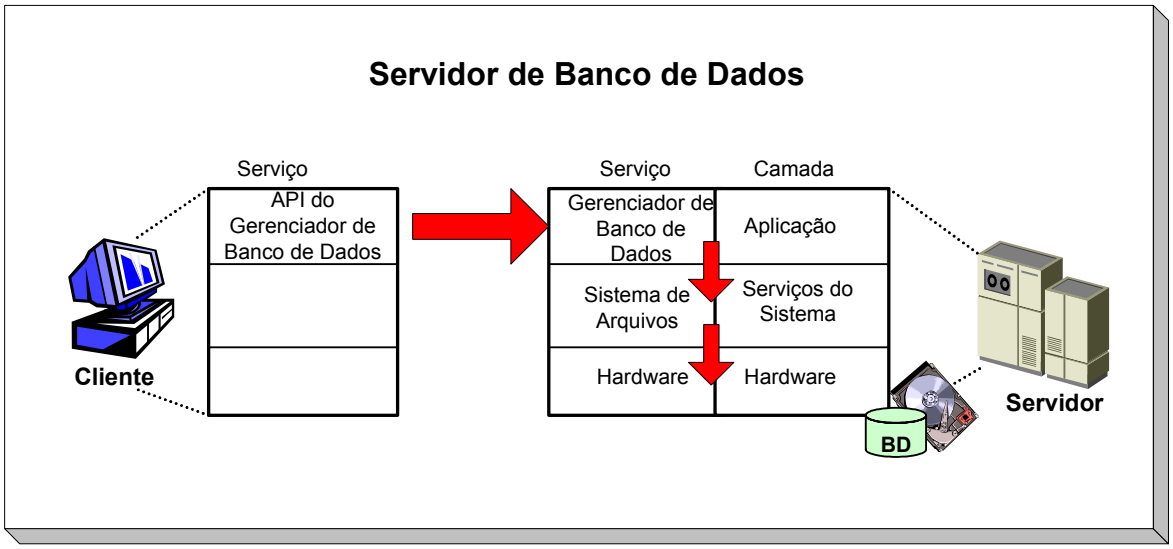

Figura 2.15 – Arquitetura C/S como Servidor de Banco de Dados

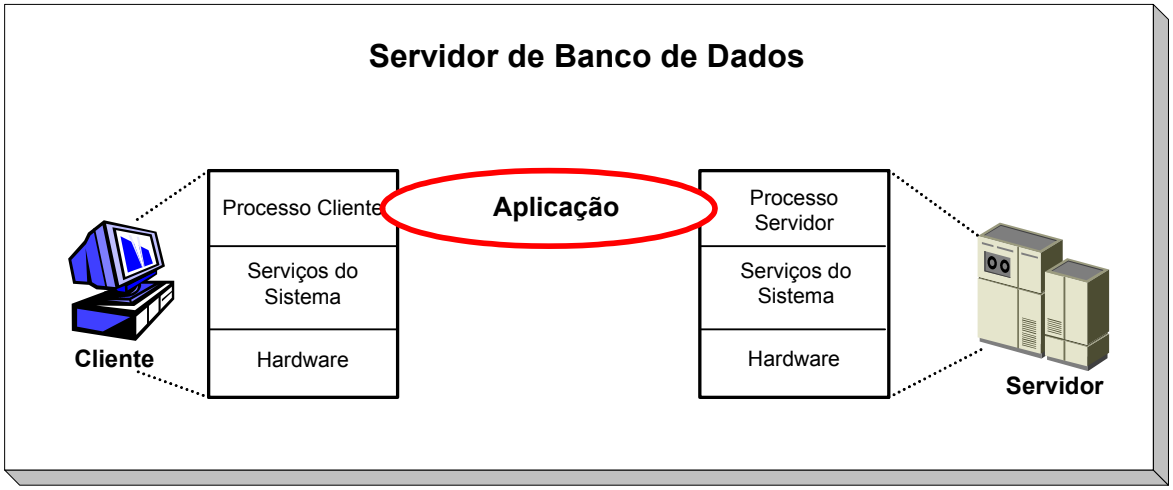

Figura 2.16 – Integração entre os Processos Cliente/Servidor

Este tipo de estrutura favorece o aumento da performance da rede de comunicação, possibilitando assim, um maior número de ligações simultâneas de diversos Clientes com diversos Servidores [28].

Embora a arquitetura Cliente/Servidor possa parecer, a princípio, uma nova versão do modelo de arquivos compartilhados através da Rede Local baseada em microcomputadores, suas vantagens são inúmeras. Fundamentalmente, ambos os modelos provêem capacidade de processamento distribuído e permitem o compartilhamento de informação, entretanto, no Modelo de Arquivos compartilhados

baseado em Servidor de Arquivos, o Cliente além de executar a aplicação, executa também o motor da Base de Dados, que por sua vez acessa essas Bases de Dados remotamente como se fossem locais.

O Servidor de Arquivos envia arquivos inteiros através da rede para o Cliente processar localmente, ocasionando congestionamento na rede, enquanto que, no modelo Cliente/Servidor, o Cliente executa parte da aplicação, sendo deixado para o Servidor a tarefa da administração da Base de Dados, comumente exercida por algum SGBD[1], o Servidor envia para o Cliente, através da rede, apenas o bloco de dados apropriado, como resultado da consulta efetuada pela aplicação.

Desta forma, na arquitetura Cliente/Servidor, cada Servidor pode suportar um número maior de usuários, uma vez que é o Cliente que gerencia a aplicação e a interface com o usuário, além do mais, com a crescente conectividade entre máquinas e sistemas operacionais, pode-se escolher para Cliente o ambiente de software e hardware que melhor se adeque as necessidades de cada aplicação do usuário, sem ter que se preocupar com o Servidor. A melhor divisão de tarefas entre o Cliente e o Servidor depende de cada aplicação em si.

Se o Servidor for apenas um SGBD, deixando para o Cliente o resto do processamento ou, se algumas tarefas como controle de acesso, análise dos dados e validação dos comandos é executada pelo Servidor. Esta é uma decisão do construtor do Sistema em função das características do negócio do usuário. As diferenças serão mostradas no tópico a seguir.

2.5 Sistema de Três Camadas para a Aplicação

Os Sistemas Cliente/Servidor têm sido utilizados basicamente nos Sistemas de Banco de Dados Distribuídos, Processamento de Transações e nos Sistemas de Suporte a Decisão, a maioria das aplicações tidas como críticas têm permanecido num *mainframe*. Pode-se definir como críticas aquelas aplicações cujos resultados são utilizados para decisões estratégicas e que, portanto, variam de empresa para empresa, dependendo do seu negócio. Quase que universalmente, os sistemas de finanças são considerados críticos para todas as empresas. Por outro lado, os sistemas de processamento de transações são críticos para as companhias de aviação.

As aplicações de processamento de transações, apesar de serem tipicamente consideradas como críticas, têm sido freqüentemente migradas para arquitetura Cliente/Servidor, através do processo de "*Downsizing*", como por exemplo os sistemas de entrada de pedidos, pontos de vendas e sistemas de

[1] SGBD: Sistema Gerenciador de Banco de Dados

reservas. Deste modo as transações ocorrem nos Clientes e são formatadas e enviadas ao Servidor para armazenamento.

Seria razoável imaginar que as primeiras aplicações a serem desenvolvidas em Sistemas Cliente/Servidor seriam relativamente simples, garantidas e não críticas, porém, as empresas que se decidiram pelo modelo Cliente/Servidor testaram-no com aplicações de processamento de transações e na maioria dos casos ficaram satisfeitas com os resultados.

Como pôde ser visto pela Figura 2.8, o *Gartner Group* apresenta cinco maneiras de se implementar a arquitetura Cliente/Servidor. Cada camada é dividida entre a parte lógica e a parte de gestão. A primeira representação refere-se a distribuição da camada de Apresentação.

2.5.1 Apresentação Distribuída

Este modelo emula uma arquitetura Cliente/Servidor. Toda a gerência da Apresentação é efetuada no Servidor, enquanto que no Cliente, somente a lógica da impressão dos caracteres no monitor é executada, como pode ser visto na Figura 2.17. O Cliente não possui qualquer tipo de "inteligência".

Existe uma técnica de tratamento cooperativo, denominada de *Revamping*, que é utilizado pela Apresentação Distribuída. O *Revamping* é uma técnica de tratamento cooperativo que está dividida em três tipos: simples, evoluído e o modificado [15].

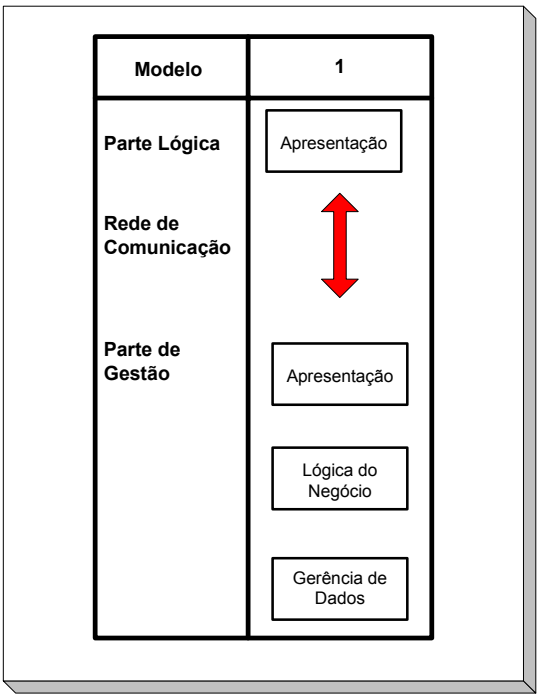

Figura 2.17 – Apresentação Distribuída

2.5.1.1 *Revamping* Simples

Consiste em habilitar as telas de modo gráfico das aplicações centralizadas com diálogos em modo gráfico. Cada mapa de tela, gerado corresponde a uma janela de interface gráfica.

2.5.1.2 *Revamping* Evoluído

Este tipo não emite simplesmente uma réplica gráfica da janela a ser criada de uma aplicação não gráfica. Constrói uma aplicação inteiramente nova, inteiramente diferente da aplicação original. Assim, um diálogo na nova aplicação pode representar um empilhamento de várias janelas da aplicação centralizada.

2.5.1.3 *Revamping* Modificado

Este terceiro tipo se deriva do segundo, porém com o fato interessante de se poder manipular a aplicação centralizada instalando sistemas gráficos, visando retirar o melhor de sua performance.

2.5.2 Apresentação Remota

O segundo modelo é definido como Apresentação Remota e possui a implementação tanto do módulo de gestão, quanto o de lógica na estação Cliente, conforme Figura 2.18, ficando este responsável (entre outras tarefas), pela manipulação das telas e pelas críticas dos dados que estão sendo inseridos.

Um ponto a ser observado, é quando se utiliza um Servidor *X-Windows* : embora o usuário se localize no Terminal *X-Windows*, ele realmente está se utilizando do Servidor *X-Windows*. O terminal real Cliente é onde o usuário está conectado. Esta característica é muito encontrada em ambientes UNIX. O Servidor *X-Windows* fica responsável pela manipulação das telas e interação com o usuário.

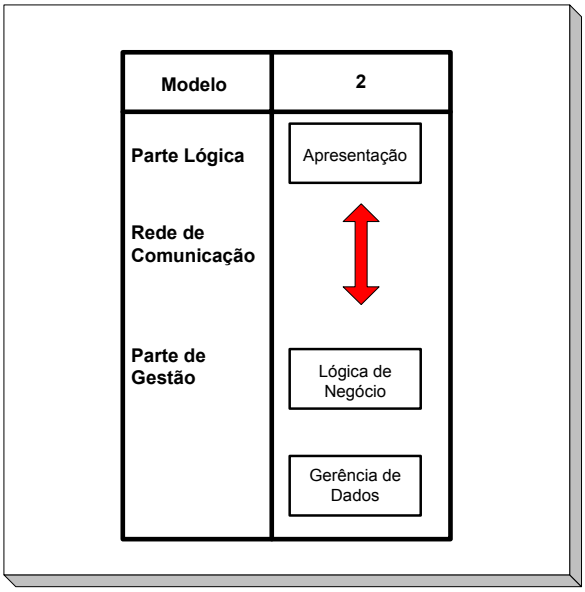

Figura 2.18 – Apresentação Remota

2.5.3 Lógica Distribuída

Outra representação é o modelo de Lógica Distribuída. Ele representa o melhor modelo de implementação para arquitetura Cliente/Servidor representado na Figura 2.19, nele, o balanço entre os processamentos clientes e servidores são bem determinados.

Neste modelo, a implantação de procedimentos armazenados (*Stored Procedures*) facilitam a performance na rede de comunicação, uma vez que somente o nome do procedimento armazenado no Servidor é transmitido pela rede.

Um ponto muito positivo para este tipo de estrutura se refere a pré-compilação necessária para os procedimentos armazenados aumentando consideravelmente o tempo de resposta. Quando os procedimentos armazenados não são utilizados, é necessário que sejam feitas compilações das sentenças das consultas antes de executá-las, evidenciando uma perda desnecessária no tempo de processamento para enviar a resposta ao Cliente. Porém, um ponto negativo para este tipo de arquitetura, é o fato de se ter que criar antecipadamente as consultas a serem utilizadas pelo sistema.

Ao contrário, se o sistema utilizar interações com o usuário este tipo de arquitetura não é o mais recomendado, pois não se podem prever todos os tipos de consultas que os usuários possam a vir a fazer.

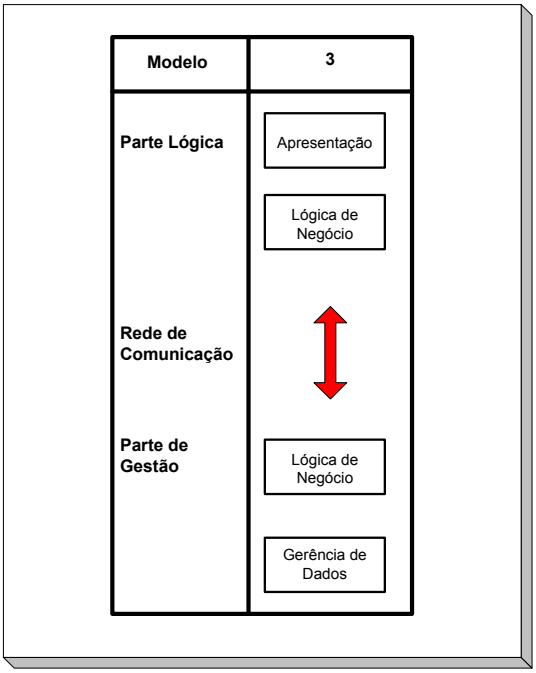

Figura 2.19 – Lógica Distribuída

2.5.4 Gerenciamento de Dados Centralizado

Neste modelo, apresentado através da Figura 2.20, toda a parte de gestão e lógica da Aplicação fica destinada ao Cliente, enquanto que somente o responsável por prover o armazenamento e a persistência dos dados permanece no Servidor. Um exemplo típico para este modelo é o caso de um SGBD, responsável direto pela gestão e lógica dos dados armazenados.

Este tipo de arquitetura é a mais difundida, não criando, teoricamente, qualquer tipo de empecilho na hora da migração de plataformas *mainframes* para plataformas *desktop* do tipo PC (*Personal Computer*).

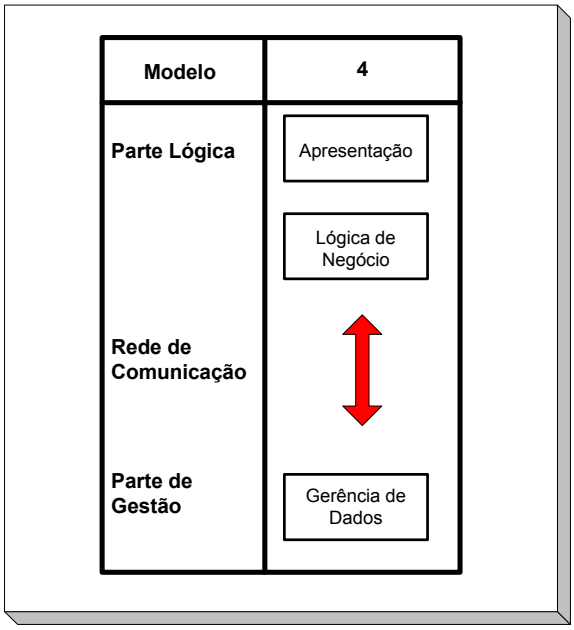

Figura 2.20 – Gerenciamento de dados Centralizado

2.5.5 Gerenciamento de Dados Distribuídos

Ao contrário do Gerenciamento de Dados Centralizados, o Gerenciamento de Dados Distribuídos, mostrado através da Figura 2.2, permite a replicação e divisão dos dados por diversos sítios (*sites*). Os sistemas distribuídos não deixam de ser micro-sistemas centralizados, necessitando, porém de estruturas como metadados para determinar a localização dos dados e suas replicações.

Já existem no mercado sistemas que se instalam entre a parte lógica da aplicação e a gerência de dados, com a função de controlar a localização real dos dados.

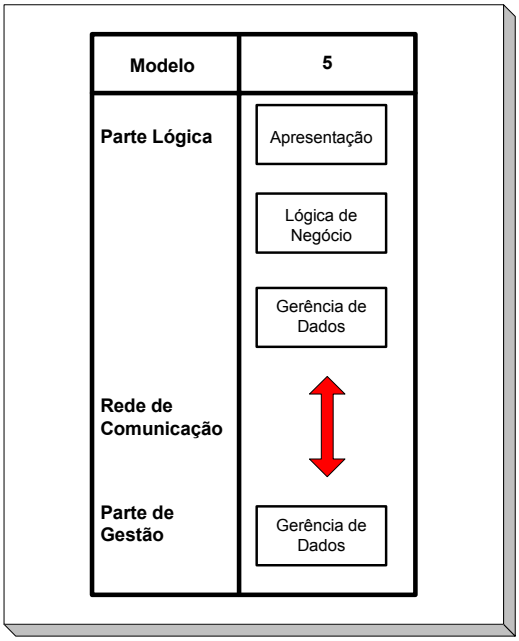

Figura 2.21 – Gerenciamento de Dados Distribuídos

Outro ponto muito importante a ser verificado é o gerenciamento da concorrência entre processos, quase sempre a cargo dos Sistemas Operacionais das plataformas, tais como o *Windows NT* e o UNIX [09] [26] [35]. O trabalho de gerenciamento para o compartilhamento dos dados se deve exclusivamente aos SGBDs.

Os tipos de arquiteturas apresentadas não são mutuamente exclusivas, mas sim complementares. É possível fazer vários tipos de associações entre Servidores e Clientes, dentro de uma rede de computadores.

A Tabela 2.1 mostra um resumo das características de sistemas Cliente/Servidor.

Atributo	**Cliente**	**Servidor**
Modo	Ativo	Reativo
Execução	Início e final fixos	Roda o tempo todo
Finalidade Principal	1. Manipulação de tela / janela 2. Interpretação de menu / comando 3. Entrada de mouse / teclado 4. Entrada de dados e validação 5. Processamento de ajuda 6. Recuperação de erro	1. Oferecer serviços funcionais 2. Compartilhamento de dados na aplicação 3. Compartilhamento de dispositivos

Atributo	Cliente	Servidor
Transparência	Oculta rede e servidores	Oculta detalhes de implementação dos serviços
Inclui	Comunicação com diferentes servidores	Comunicação com diferentes clientes
Exclui	Comunicação Cliente – Cliente	Comunicação Servidor – Servidor

Tabela 2.1 – Principais Tópicos de uma Arquitetura Cliente/Servidor

Conclusão

Os modelos da arquitetura cliente/servidor propõem maneiras de organizar as partes integrantes na comunicação entre computadores, definindo as atribuições em cada elemento participante e o fluxo da informação da origem ao seu destino. O conhecimento dos processos e das camadas da arquitetura em cada um destes elementos fornecem informações fundamentais para a elaboração de aplicações mais robustas e eficientes.

Capítulo 3

O cenário atual de execução das aplicações

O objetivo neste capítulo é bem amplo e técnico, porém essencial. As aplicações têm sido desenvolvidas para suprir principalmente o mercado comercial, automatizando e informatizando processos de trabalho.

Devido a grande diversidade de áreas de aplicação, passou-se a utilizar uma série de nomenclaturas e títulos para diferenciar os aplicativos, incluindo novas siglas que merecem aqui serem estudadas, porém não poderíamos esquecer o cenário atual utilizado por estes aplicativos.

A arquitetura cliente/servidor e o protocolo TCP/IP fornecem uma série de mecanismos que podem comprometer a segurança e o sigilo da informação, entretanto existe uma série de outros protocolos que podem ser utilizados nas redes locais em conjunto com o TCP/IP.

Com ferramentas automatizadas e disponíveis através da Internet, atacantes aproveitando-se das falhas existentes no cenário da comunicação entre as partes cliente/servidor de uma aplicação, podem ter acesso as informações trocadas através da rede ou mesmo ter acesso aos computadores e servidores envolvidos na comunicação.

A exposição destes protocolos e os mecanismos utilizados para comunicação são essenciais para compreendermos quais os pontos vulneráveis que são utilizados para a realização de ataques.

3.1 Característica dos protocolos de comunicação

Não são apenas os erros de programadores que causam o problema de segurança que vemos nas aplicações. Existe um conjunto de fatores que, caso não recebam a devida atenção, podem proporcionar a exposição da informação que trafega numa rede de computadores. Dentre os quais podemos destacar o Sistema Operacional utilizado, a aplicação de correções de segurança por parte dos administradores destes sistemas, o conhecimento da tecnologia de rede empregada, suas características, topologias e principalmente os protocolos de comunicação utilizados.

Fatores de mercado também influenciaram na construção das aplicações. O grande crescimento da Internet –que permitiu a integração a custos menores de redes geograficamente distantes, o barateamento e o surgimento de novos componentes para a construção de redes locais e, paralelamente o domínio da *Microsoft* com *MS-Windows* em estações de trabalho nas empresas e computadores residenciais, tornou praticamente obrigatória a construção e adaptação dos softwares para esta nova realidade.

Em virtude dos fatos apresentados, dois protocolos difundiram-se rapidamente, são eles o NetBIOS® (que posteriormente seria substituído pelo NetBEUI) protocolo padrão das redes *Microsoft* e o TCP/IP utilizado como padrão nas estações UNIX e também como padrão na Internet.

As vulnerabilidades encontradas nestes protocolos, somadas ao conjunto de fatores descritos, proporcionam a atacantes realizarem suas investidas e terem acesso as informações importantes que trafegam na rede. Sendo assim é necessário um estudo detalhado destes protocolos, bem como dos tipos de aplicações existentes no mercado que utilizam esta tecnologia.

Antes de iniciar sobre este assunto é importante conhecer quais as características principais dos softwares comerciais existentes, suas denominações e tendências.

3.1.1 Aplicativos prontos ou sob medida

Os aplicativos podem ser separados em dois grandes grupos: os prontos e os feitos sob medida [17]. Os aplicativos comerciais prontos são vendidos na forma de “pacotes”, com vários programas prontos para instalação e uso. Existem pacotes para todos os tipos de atividades.

O usuário pode escolher o mais adequado para suas necessidades e instalá-lo sem grandes dificuldades. A vantagem está no fato de que como o produto é vendido em larga escala, os custos de desenvolvimento são rateados entre milhares de compradores e o preço final pode ser mais barato, por

outro lado, como os pacotes são padronizados, podem não atender completamente às suas necessidades e talvez o usuário não ache um produto que funcione exatamente como deseja.

Aplicativos feitos sob encomenda são usados, geralmente, por grandes organizações, que podem manter equipes internas de programadores para resolver suas necessidades específicas.

Organizações que precisam de aplicativos especiais, mas não podem ou não desejam manter equipes internas de programadores, costumam recorrer a *software-houses*, empresas que desenvolvem aplicativos sob encomenda, deste modo aplicativos feitos sob encomenda podem ser ajustados exatamente as necessidades de uma organização, porém tem como desvantagem manter o investimento e os altos custos de desenvolvimento, como a manutenção da equipe de programadores e a elaboração da documentação.

Existem outras formas de distribuição de aplicativos:

- *Shareware* – Um programador ou uma pequena empresa de programação cria um aplicativo e o disponibiliza gratuitamente, no todo ou em parte, via Internet ou na forma de disquetes. O usuário pode experimentar o programa e só terá de pagar se decidir adotá-lo para seu uso regular. Geralmente o preço é baixo, pois não inclui gastos com embalagem, distribuição e publicidade.

- Venda ou aluguel de módulos – Os usuários podem utilizar módulos disponibilizados via Internet, pagando pelo tempo de uso. O custo é bem baixo, já que a distribuição via Internet elimina muitos gastos. Os programas são divididos em componentes específicos para cada tarefa a ser realizada e o usuário sempre tem acesso a versão mais atualizada de cada programa, esse esquema ainda é experimental, mas sua importância tende a aumentar com o desenvolvimento de novas tecnologias.

3.1.2 Aplicativos avançados e tendências

Além dos aplicativos que estamos acostumados a ver em ação em computadores domésticos e na maioria das organizações, existem outros que utilizam recursos mais sofisticados e avançados. O uso desses programas é restrito, seja por seu elevado custo, seja pela necessidade de pessoal especializado ou, ainda, pelo poderoso hardware necessário para executá-los [17].

Muitos dos produtos que utilizamos diariamente, como equipamentos eletrônicos, têm sua fabricação controlada por esses aplicativos. O mesmo acontece com serviços essenciais, como fornecimento de luz elétrica e controle de tráfego aéreo.

3.1.2.1 *Business Intelligence*

Business Intelligence (BI) é um conjunto de conceitos e métodos que, fazendo uso de acontecimentos (fatos) e sistemas baseados em acontecimentos, apóia a tomada de decisões. Há produtos de BI desde a década dos 70, cuja característica era a grande programação que exigiam e os altos custos de implantação.

Com o surgimento dos bancos de dados relacionais, dos microcomputadores pessoais e das interfaces gráficas (como *Windows*, OS2 dentre outros), aliados ao aumento da complexidade dos negócios, começaram a surgir os primeiros produtos realmente direcionados aos analistas de negócios.

Os sistemas de BI têm como características:

- extrair e integrar dados de múltiplas fontes;
- fazer uso da experiência;
- analisar dados contextualizados;
- trabalhar com hipóteses;
- procurar relações de causa e efeito;
- transformar os registros obtidos em informação útil para o conhecimento empresarial.

Existe no mercado uma série de ferramentas para BI, as mais importantes estão relacionadas com a extração e tratamento dos dados de bancos de dados relacionais. Em geral os dados são exportados para interfaces amigáveis para análise mais crítica da informação, desta maneira pode-se considerar como ferramentas de BI produtos que tenhas as seguintes características:

- Planilha eletrônica;
- Geradores de pesquisa e de relatórios;
- DSS – *Decision Support System* (Sistema de Suporte à Decisão);
- EIS – *Executive Information System* (Sistema de Informação Executiva);
- *Data Warehouse*;
- *Data Mart*;
- OLAP – *Online Analytical Processing* (Processamento Analítico *On-Line*);
- *Data Mining*.

3.1.2.2 Sistemas de Gestão Empresarial

Essas ferramentas são concebidas para integrar todas ou, quase todas, as principais funções na organização (como contabilidade, produção, vendas, compras, distribuição) num único sistema, usado

para operar e gerenciar toda a organização, em geral, são projetados para necessidades bem específicas e contam com poderosos bancos para o armazenamento e a manipulação dos dados. A implementação costuma ser muito cara e, para que tudo dê certo, é importante que os usuários sejam envolvidos no projeto. Além disso, eles devem receber treinamento para usar bem o sistema [17].

3.1.2.3 EIS – *Executive Information System*

Os EIS (*Executive Information Systems*) – Sistemas de Informações Executivas – são sistemas desenvolvidos para atender as necessidades dos executivos de uma organização. O objetivo principal é obter informações gerenciais de uma maneira simples e rápida. Ele é concebido conforme o nível de conhecimento, entendimento e compreensão de informática dos diretores.

Através do EIS, o executivo obtêm de forma simples e rápida as informações gerenciais para que possa tomar suas decisões, sem depender exclusivamente do corpo técnico ou da área de informática. O intuito desse sistema é facilitar a vida desses profissionais. Geralmente, possuem interface gráfica, são simples de operar, apresentam resultados de pesquisas no formato de gráficos e não em tabelas como a maioria dos outros sistemas, de forma que a visualização dos dados seja mais clara.

Devido a falta de tempo desses usuários especiais, o EIS deve ser modelado para ser o mais amigável possível. As informações devem ser acessadas, de preferência, com poucos cliques. Outra característica importante é que as informações devem ser trazidas num nível bastante resumido e altamente agregadas, pois as decisões tomadas nesse nível administrativo não se atêm aos detalhes e sim ao todo. Ao diretor de uma organização de grande porte interessa mais conhecer o valor total das despesas de material de expediente, por exemplo, do que saber quantas canetas foram usadas durante um ano.

Pode-se construir o EIS utilizando-se como base vários sistemas transacionais existentes na organização, mas isso pode tornar o trabalho inviável, devido ao grande volume de tarefas que se gerariam com a extração de dados de diversas fontes, tratamento dado a eles e a manutenção necessária para garantir a integridade e o histórico dos dados.

Atualmente a tendência é que estes EIS acessem um *Data Warehouse*, que é uma combinação de diferentes bases de dados existentes na instituição e organizados de forma a possibilitar a tomada de decisão. A tarefa fica muito mais fácil, pois a busca dos dados pode ser feita num único sistema, onde as informações já se encontram formatadas de acordo com as necessidades dos executivos. Como a

tarefa de extração, transformação e carga dos dados fica por conta dos técnicos do *Data Warehouse*, o executivo deve ocupar-se, apenas, da análise dos dados.

As principais características que os EIS devem apresentar são:

- Ter como propósito atender às necessidades de informações dos executivos de alto nível: acompanhamento e controle de informações a nível estratégico da organização;
- Poder ser customizados de acordo com o estilo de cada executivo;
- Dispor de recursos gráficos de alta qualidade para que as informações possam ser apresentadas graficamente de várias formas;
- Ser fáceis de usar, para que o executivo possa operá-los com muito pouco treinamento;
- Proporcionar acesso rápido e fácil a informações detalhadas; ou seja, as informações são visualizadas em níveis que podem ser expandidos.

O EIS além de facilitar o trabalho num ambiente fácil e simples, também deve permitir que o usuário altere o nível de detalhamento das informações trabalhadas, permite que o usuário faça análises de tendências e comparações entre diversas informações de forma prática e rápida, sem necessitar de um apoio da área de informática para conseguir atingir o seu objetivo [17]. Isso torna os executivos mais competitivos, organizados e independentes, que é tudo o que eles querem.

3.1.2.4 Softwares integrados de gestão (ERP)

A sigla ERP vem de "*Enterprise Resource Planning*" (Planejamento dos Recursos Empresariais) e designa pacotes integrados de gestão. Esses pacotes integrados são sistemas de informação com módulos integrados que dão suporte a diversas áreas operacionais da organização, tais como vendas, produção, gestão de materiais, contabilidade e pessoal. Essa integração reduz a quantidade de programas e fornecedores diferentes e permite a implantação de um banco de dados único, compartilhado por todas as aplicações.

Um ERP permite que processos organizacionais sejam realizados de forma integrada, ultrapassando os limites departamentais e automatizando ao máximo as atividades [17], além da promessa de reduzir custos de TI com pessoal, desenvolvimento de software, treinamento e manutenção, existem outros benefícios associados ao uso de pacotes integrados de gestão:

- A integridade dos dados corporativos fica protegida;
- A disponibilidade imediata de todos os dados para todos os usuários que necessitem e tenham autorização de acesso;

- A maior flexibilidade na modificação dos fluxos de processos organizacionais, dentro da gama de soluções oferecida pelo fornecedor do pacote;
- A visão sistêmica dos eventos e objetos organizacionais.

Os principais fabricantes de software integrado de gestão no mercado são, dentre outros, *SAP*, *Oracle*, *PeopleSoft* e *BAAN*.

A opção por um pacote padrão, composto por softwares integrados, que substitua os sistemas corporativos tradicionais, tem se tornando cada vez mais atrativa, principalmente em razão do custo elevado de desenvolvimento e manutenção de softwares produzidos internamente ou sob encomenda. Muitas organizações implementaram ERP, pois desejavam substituir sistemas que estavam ficando obsoletos. Já em outras organizações, a integração de sistemas era o ponto crítico – fusões e aquisições deixaram-nas com uma coleção de sistemas ineficientes, não confiáveis e incompatíveis.

A maioria das organizações tem a expectativa de que a adoção de um ERP reduza os seus custos de operação. Elas também esperam que o sistema produza melhorias em processos específicos, tais como os de logística, planejamento da produção e atendimento ao cliente, entretanto muitas organizações estão descobrindo do modo mais difícil que a integração maciça de aplicações é algo mais complexo do que esperavam.

O caminho da implantação passa por cinco estágios: projeto, implementação, estabilização, melhoria contínua e transformação.

3.2 O cenário de comunicação das aplicações

Alguns anos atrás havia um cenário de incerteza quanto a padronização da comunicação. Muitas aplicações eram desenvolvidas para serem executadas sobre redes proprietárias, com protocolos específicos. Com o passar do tempo, a arquitetura TCP/IP tornou-se um padrão de mercado e qualquer aplicação que pretende ter sucesso no meio comercial e acadêmico deve ter um suporte a este protocolo.

3.2.1 Redes de Computadores e protocolos

Normalmente várias perguntas são feitas para se tentar entender como clientes e servidores se comunicam, que tipo de protocolo é utilizado e como um cliente encontra o servidor desejado dentro de uma rede de computadores. Embora estas perguntas possam parecer muito difíceis, na realidade elas são relativamente fáceis. Existem vários conceitos e aspectos comuns as comunicações Cliente/Servidor necessários ao entendimento do funcionamento da comunicação.

Nesta seção será dada uma visão geral sobre os conceitos básicos de protocolos, passagem de mensagem, os aspectos de conexão e sincronismo de processo.

Há duas décadas, diversos esforços vêm sendo traçados para se estabelecer um padrão único para redes de computadores. No entanto, foram definidos não apenas um, mas vários modelos de referência. Aqui serão expostos dois principais, o modelo de referência OSI/ISO e o TCP/IP.

3.2.1.1 O Modelo de Referência OSI/ISO

Em março de 1977, foi constituído pela ISO[1] um grupo de trabalho para estudar a padronização da interconexão de sistemas de computação. Foi definida, inicialmente, uma arquitetura geral, denominada MODELO DE REFERÊNCIA OSI[2], para servir de base para a padronização da conexão de sistemas abertos [37] [36].

O modelo OSI possui sete camadas, como apresentado na Figura 3.1. Cada camada representa um grupo específico de tarefas de comunicação. A norma descreve o escopo funcional de cada camada e os requisitos para a interface com as camadas adjacentes (serviços).

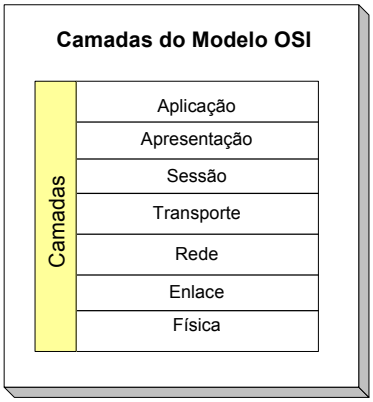

Figura 3.1 – Camadas do Modelo de Referência OSI

A Figura 3.2 mostra que quando a mensagem passa da camada *n+1* para a camada *n* são acrescidos outros dados relevantes a camada *n*. Estes dados são retirados quando a mensagem chega na camada de mesmo nível na estação destino. Estes acréscimos podem ser informações tais como: tipo da mensagem, endereços, tamanho da mensagem, código de detecção de erro entre outros.

[1] ISO: *International Organization for Standardization*
[2] OSI: *Open System Interconnection*

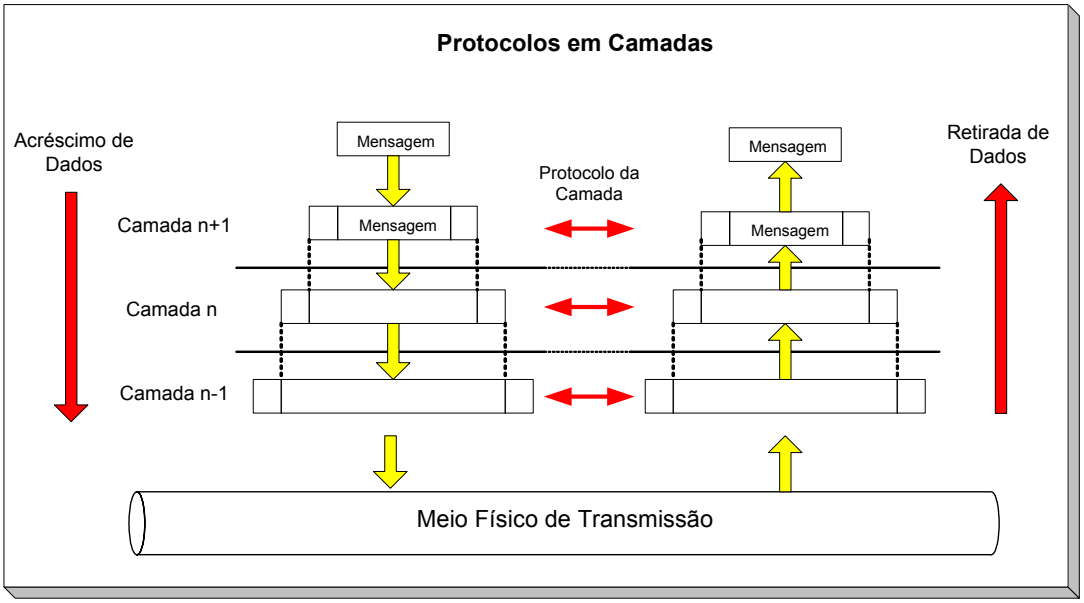

Figura 3.2 – Arquitetura de protocolos em camadas

O modelo OSI/ISO especifica todas as primitivas de comunicação para que haja troca de mensagens entre as camadas [01] [06] [11] [28][36]. Cada camada adiciona um cabeçalho para que haja uma identificação na máquina destino. A Figura 3.3 baseada em [44] [45], mostra os principais protocolos usados nas comunicações Cliente / Servidor baseados na *Microsoft*, Internet e IBM.

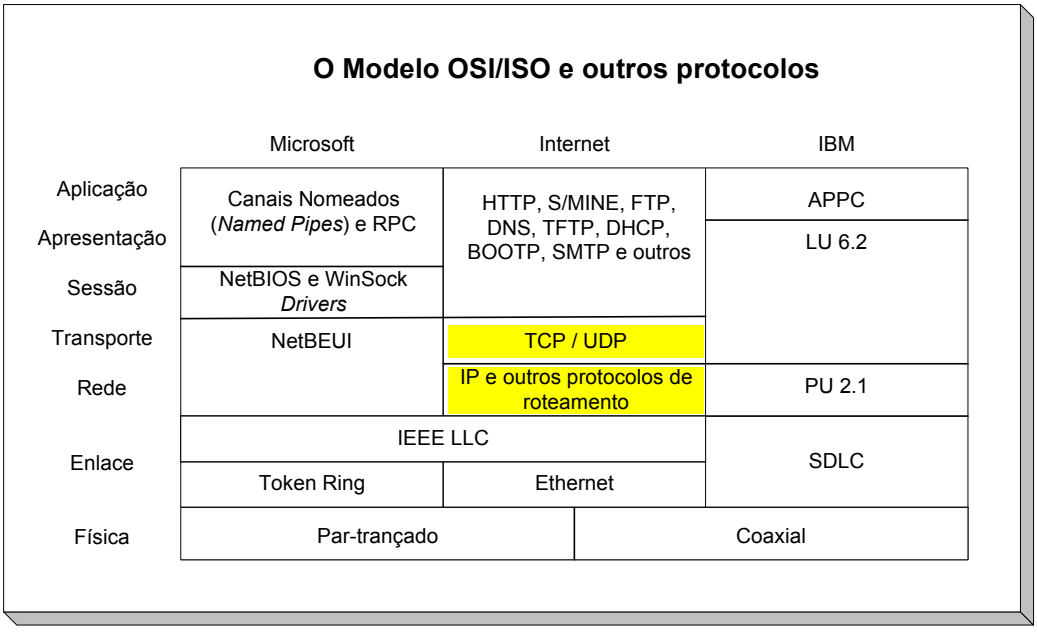

Figura 3.3 – Comparação entre o modelo OSI/ISO e outros protocolos

Tendo em vista os objetivos do trabalho pretendido pela presente tese, somente o protocolo de comunicação TCP/IP e alguns outros que forem essenciais para o desenvolvimento deste trabalho serão descritos.

3.2.1.2 O Protocolo TCP/IP

O TCP/IP distingue-se dos demais protocolos pelo seu endereçamento universal. Cada máquina em uma rede possui um endereço que a identifica. A camada TCP é orientada à conexão, enquanto a camada IP trabalha sem conexão [06] [10] [11]. Cabe a camada IP o trabalho de distribuir os datagramas entre as máquinas de uma rede. Para fazer este serviço, ele possui um único endereço de 32 bits que contém informações suficientes para identificar univocamente uma rede e um determinado computador. Este endereço é comumente escrito em decimal de quatro bytes separados por um ponto. Por exemplo:

10001111, 01101010, 00001010, 00000001, representam o endereço 143.106.10.1.

Existem 5 classes onde estes endereços são divididos. A Figura 3.4 mostra estas classificações.[27] [28]

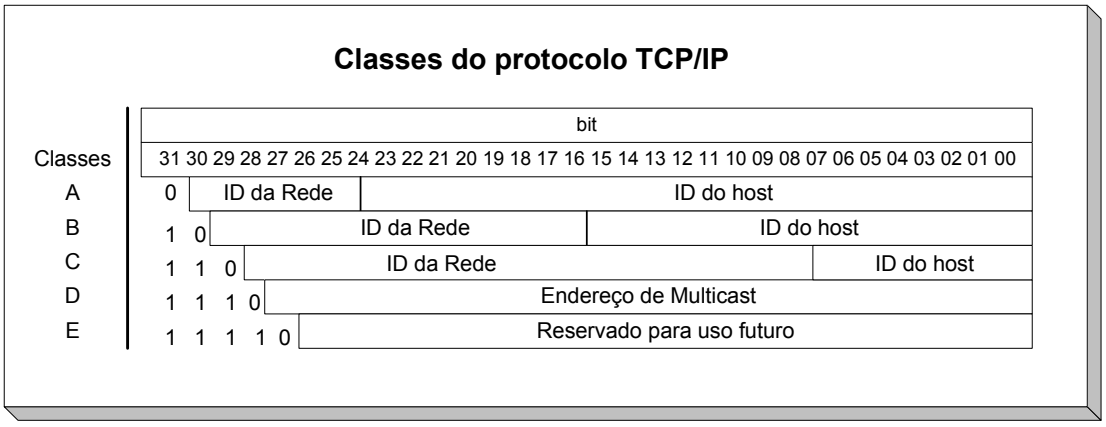

Figura 3.4 – Classes do Protocolo TCP/IP

O número de redes e *hosts* que podem ser endereçados pelas Classes A, B, C, D e E, bem como o escopo de seus endereços IP são definidos na Tabela 3.1. Por exemplo, é possível endereçar até 65.136 *hosts* para cada rede da classe B. Portanto, 65.136 x 16.382 = 1.067.057.952.

Isto quer dizer que mais de 1 milhão de *hosts* podem ser endereçáveis pela classe B, já que certos endereços IP não podem ser utilizados por serem destinados a trabalhos específicos, como o endereçamento de *Gateways* e de *Broadcasting*.

Classes	Nº de Redes	Endereços de Rede	Nº de Máquinas
A	128	0.0.0.0 – 127.255.255.255	16.777.216
B	16.384	128.0.0.0 – 191.255.255.255	65.536
C	2.097.152	192.0.0.0 – 223.255.255.255	256
D	*Multicast*	224.0.0.0 – 239.255.255.255	
E	Reservado	240.0.0.0 – 247.255.255.255	

Tabela 3.1 – Classes e endereçamento em redes TCP/IP

Com isto, quando se tem um endereço de rede na classe B e deseja-se aumentar o número de redes possíveis, em detrimento da capacidade de máquinas que podem ser instaladas, ou vice-versa, utiliza-se a técnica da máscara de rede (*NetMask*). Esta técnica permite a criação de sub-redes através da modificação local da linha divisória que separa os bits de identificação das redes e dos *hosts*. Cada sub-rede também é representada pela notação de ponto decimal. Os bits da *NetMask* definidos como "um" identificam a porção da rede, enquanto os definidos por "zero", representam os *hosts*.

Por exemplo, caso uma empresa possua o endereço IP 166.78.4.139 da classe B, isto indica que o computador reside na rede 166.78 e que tem uma identificação de *host* 4.139. Porém, caso os administradores da rede decidam utilizar a *NetMask* 255.255.255.0 para aumentar o número de redes internas a empresa, é necessário que se faça uma combinação AND, do endereço IP original com a *NetMask*, alterando a identificação de rede para 166.78.4 e a identificação de *host* para 139. Pode-se dizer com isso, que o computador é o *host* 139 na sub-rede 166.78.4.0 (endereço de *gateway*).

3.2.1.3 O Protocolo SMB

SMB (*Server Message Block Protocol*) – É um protocolo de nível de aplicação utilizado por Sistemas Operacionais como *Windows* for *Workgroups*, *Windows 95/98/2000/2003*, *LAN Manager and Windows NT and by IBM's OS/2* e *LAN Server* [13].

Este protocolo é utilizado para implementar o controle de sessão de rede, além de prover um mecanismo para o compartilhamento de arquivos, impressoras, portas seriais entre outros recursos dos computadores que estão atuando como servidores destes recursos. Possui funcionalidade análoga aos *AppleTalk Session Protocol*, *AppleTalk Filing Protocol*, *Printer Access Protocol* no conjunto de protocolos da *Apple* [13].

O protocolo SMB funciona baseado em uma arquitetura cliente/servidor, onde os clientes enviam requisições para a utilização dos recursos disponíveis nos servidores e estes por sua vez respondem ou, permitindo o acesso aos recursos ou impedindo este acesso, dependendo do nível de segurança atribuído aos recursos requisitados. A segurança no protocolo SMB é provido em dois níveis [13]:

Nível de compartilhamento – Neste nível uma senha é atribuída a cada recurso a ser compartilhado e o cliente só pode acessar este recurso se tiver posse desta senha, entretanto uma vez que o cliente ganha acesso ao recurso ele pode acessar todos os arquivos presentes neste recurso compartilhado.

Nível de usuário – No nível de usuário é possível atribuir senhas para arquivos individuais e cada usuário poderá ter um privilégio diferente de acesso ao arquivo. Aqui o usuário deve primeiro se autenticar no servidor e ser autenticado por este servidor, ao fazer isto para este usuário, é associado um Identificador Único (**UID**), utilizado para o gerenciamento de acesso aos recursos compartilhados.

Para efetuar a conexão com os servidores, o SMB necessita de protocolos de transporte e mais comumente utiliza uma das seguintes combinações: TCP/IP – mais especificamente NetBIOS sobre TCP/IP como especificado nos RFCs 1001 e 1002 -, NetBEUI ou IPX.

Depois que a conexão foi estabelecida entre os clientes e servidores, os clientes estão aptos a enviarem comandos SMB para os servidores e podem ganhar acesso ao sistema de arquivos de outras máquinas, podendo abrir, ler escrever e fazer qualquer outra operação sobre o sistema de arquivos remotamente através de comandos SMB [13]. Para anunciar a sua presença na rede os servidores enviam mensagens de broadcast de tempos em tempos. Os clientes escutam por estas mensagens e constroem uma lista de servidores disponíveis.

Quando o SMB está utilizando o NetBEUI como camada de transporte, este mecanismo funciona satisfatoriamente bem, uma vez que as mensagens ficam confinadas em um mesmo segmento de rede, entretanto quando a camada de transporte é o TCP/IP, problemas podem surgir, uma vez que os roteadores não costumam propagar mensagens de broadcast para fora da sub-rede, embora isto possa ser configurado, este tipo de artifício não combina muito bem com a arquitetura e o modelo oferecido pelo TCP/IP [13].

3.2.1.4 O Protocolo NetBIOS e NetBEUI

NetBIOS (*Network Basic Input / Output System*) – foi desenvolvido pela *Sytec Inc*. (hoje *Hughes LAN Systems*) para a IBM em 1983. É geralmente descrito como um protocolo de nível de sessão,

entretanto o NetBIOS não é um protocolo e sim uma interface que foi projetada para ser uma extensão da BIOS para oferecer serviços de rede, tais como tradução de nomes de *hosts* para endereços e transferência de dados. Resumidamente é uma maneira que as aplicações têm de se comunicar com a rede [13].

Foi implementado como uma API[1], embora com o passar do tempo passou a ser chamado de protocolo NetBIOS. Geralmente, o termo protocolo NetBIOS é utilizado para referenciar o conjunto de protocolos que utilizam a API NetBIOS e o nome oficial utilizado pela IBM é *NetBIOS Frame Protocol* (NBF) [13]. O NetBIOS criou uma interface entre os programas de aplicação e os protocolos de rede, assim aplicações podem ser escritas para acessar apenas a camada mais alta do modelo OSI, permitindo que sejam transportadas por várias tecnologias diferentes. Neste modelo qualquer aplicação pode usar a rede fazendo chamadas usuais ao Sistema Operacional e o NetBIOS habilita que o usuário acesse aplicações em outras máquinas através da rede.

Este protocolo foi desenvolvido para a utilização em redes pequenas que compartilham o mesmo meio físico e oferece dois tipos de serviço: orientado à conexão e não orientado à conexão, de forma *broadcast* e *multicast*. Quando utilizado em redes maiores com diversos segmentos ele é implementado utilizando outros protocolos que possuam capacidade de roteamento para prover o transporte, entre eles o TCP/IP. As RFCs relevantes, que descrevem como o NetBIOS é encapsulado e carregado em TCP e UDP são os seguintes [13]:

RFC 1001 - *PROTOCOL STANDARD FOR A NetBIOS SERVICE ON A TCP/UDP TRANSPORT: CONCEPTS AND METHODS*

RFC 1002 - *PROTOCOL STANDARD FOR A NetBIOS SERVICE ON A TCP/UDP TRANSPORT: DETAILED SPECIFICATIONS*

Quando utilizam o transporte baseado em TCP/UDP existem três tipos de serviços que podem ser implementados com este protocolo: O *NetBIOS Name Server* (NBNS) e o *NetBIOS Datagram Distribution Server* (NBDD) e um serviço de sessão. O NBNS pode ser configurado para trabalhar de várias maneiras. As duas mais comuns são como um quadro de avisos, onde os sistemas podem registrar os seus nomes e a segunda maneira é onde ele pode gerenciar completamente nomes e endereços. O NBDD serve para prover funcionalidade de *broadcast* e *multicast* em *Intranets*. Desta forma datagramas podem ser enviados para nós individuais ou para todos (difusão).

[1] API: *Application Program Interface*

Neste serviço as mensagens possuem comprimento de 512 bytes e a entrega não é garantida, isto é, o serviço não é seguro, mas permite a comunicação com várias máquinas ao mesmo tempo. O serviço de Sessão oferece uma maneira orientada à conexão para a comunicação entre duas estações. Nesta modalidade de serviço deve existir obrigatoriamente um cliente e um servidor, as mensagens são de 64 Kbytes e o serviço é seguro, manipulando detecção e correção de erros, mas só permitem conexões *unicast* entre duas máquinas. O NetBIOS também pode ser carregado pelo conjunto IPX/SPX da *Novell* e por *frames* PPP (*Point-to-Point*).

NetBEUI (*NetBios Extented User Interface*) – O NetBEUI é uma versão melhorada do NetBIOS. Ele formaliza o frame de transporte que não foi padronizado no NetBIOS. O NetBEUI implementa o OSI LLC2 [1][13]. Este protocolo foi o protocolo de redes original do PC[2], utilizado pela IBM em seu servidor *LanManager*. Depois foi adotado pela *Microsoft* para os produtos de rede da empresa. Ele define a maneira como as aplicações enviam e recebem mensagens sobre o NetBIOS *Frame Protocol*.

Sua especificação está em um documento da IBM chamado de "*IBM Local Area Network Technical Reference Manual*" que executa sobre a camada de enlace padrão 802.2. Ele também é um protocolo não roteável, que se tornou a maior limitação deste protocolo assim como para o NetBIOS, que deve rodar sobre um protocolo de rede e transporte para poder ser utilizado em um ambiente WAN [13].

3.2.2 Aspectos de Conexão

Existem dois tipos de conexões usados para a comunicação. O primeiro é chamado sem conexão, onde cada mensagem encontra seu próprio caminho até o seu destino [06] [11] [27]. Cada mensagem é independente de todas as outras mensagens, podendo atingir rotas diferentes até o seu destino. A Internet utiliza este tipo de conexão. Devido a esta característica, cada mensagem deve conter toda a informação de endereçamento necessária para a sua entrega. Dependendo do serviço de entrega usado, as mensagens podem chegar fora de ordem ou serem mal encaminhadas no trânsito.

As mensagens numa comunicação sem conexão são chamadas de datagramas. Essa forma de comunicação também é conhecida como comutação de pacotes. Pode-se fazer uma analogia deste tipo de comunicação com o serviço de entrega dos correios. Não existe uma rota preestabelecida quando se utiliza o envio de mensagem por meio de datagrama. A Figura 3.5 mostra os diversos caminhos que as mensagens podem percorrer.

[1] OSI: Logical Link Control, type 2.
[2] PC: Personal Computer

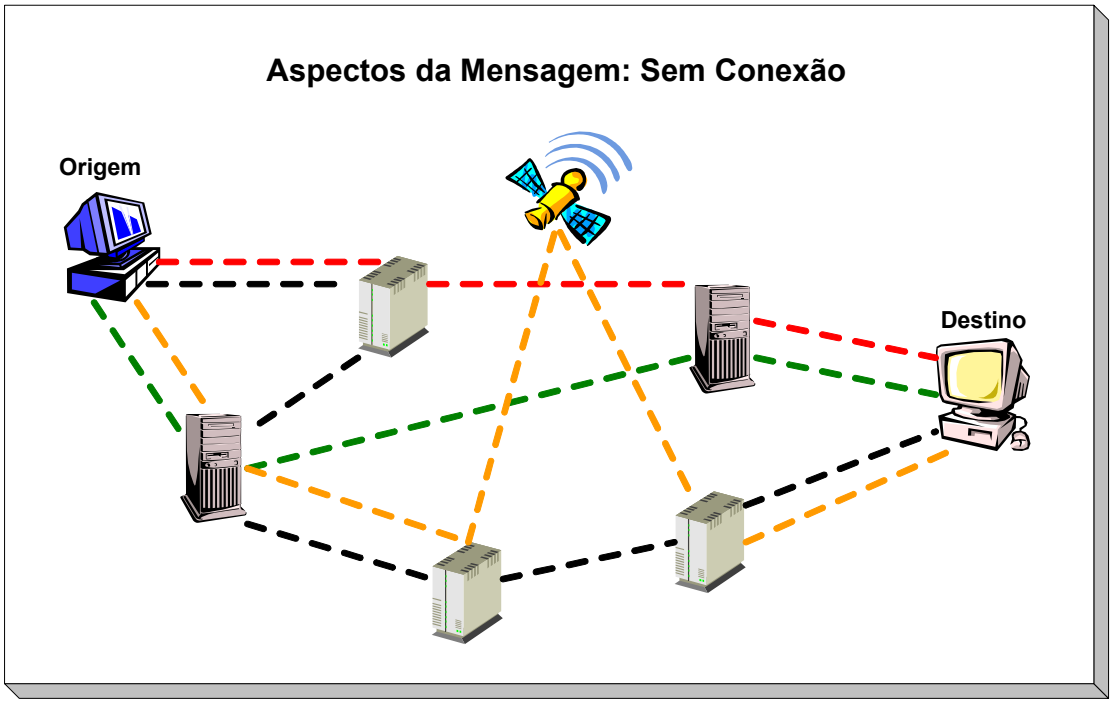

Figura 3.5 – Conexão por datagramas: sem conexão

A mensagem, que é dividida em vários pacotes, pode seguir caminhos distintos, tanto do *host* 1 (origem) para o *host* 2 (destino), quanto no sentido inverso, podendo, inclusive, chegar ao seu destino em tempos diferentes, havendo assim, a necessidade de reordenação da mensagem na parte responsável da comunicação pela aplicação do *host* de destino.

O outro tipo de comunicação é o baseado em conexão, onde um circuito prévio é estabelecido antes que a troca de dados se efetue. Quando um circuito estiver conectado, cada mensagem segue uma seqüência e sempre é direcionada ao longo do mesmo circuito [06] [11] [27]. Logo, cada mensagem só precisa de um identificador de circuito para ser direcionada para o seu destino. O recebimento de cada mensagem normalmente é confirmado e, se for o preciso, o controle de fluxo é usado para regular a velocidade com que as mensagens são enviadas.

A comunicação baseada em conexão, conforme apresentado na Figura 3.6, garante a transmissão das mensagens, estas são também divididas em pacotes, sem perda para o seqüenciamento dos pacotes nos *hosts* destinos, diminuindo assim o trabalho de implementação na parte de comunicação do aplicativo.

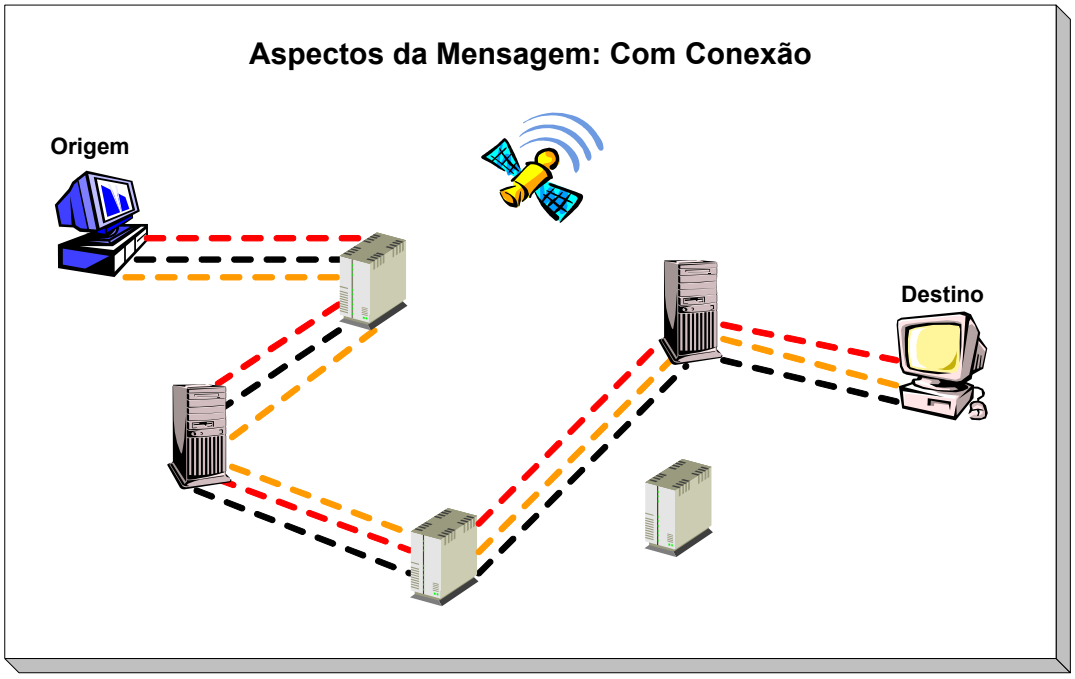

Figura 3.6 – Comunicação baseada em conexão

Quando a comunicação é finalizada, torna-se necessário desfazer a conexão para poder liberar os recursos de rede utilizados. Deve-se notar que as mensagens sempre chegam ordenadas neste tipo de conexão. Um exemplo de comunicação baseado em conexão é o sistema telefônico, na Tabela 3.2 compara os dois tipos de comunicação.

Característica	Com Conexão	Sem Conexão
Tipo de mensagem	Fluxo de Dados	Datagrama
Rota	Estática	Dinâmica
Endereçamento da Mensagem	Endereço de destino completo para estabelecer circuito, após estabelecimento necessita apenas do ID do circuito	Endereço destino completo em todas as mensagens
Confiabilidade	Seqüenciado, controle de erro e fluxo, entrega garantida	Sem garantias, as mensagens podem se perder ou chegar ao destino fora de ordem
Opções	Podem ser negociadas durante a	Não disponíveis

	preparação	
Sincronismo	Explícito	Implícito
Overhead	Preparação e liberação do circuito	Rota da mensagem

Tabela 3.2 – Tipos de Conexão

3.2.3 Aspectos de Sincronismo e Passagem de Mensagem

A comunicação entre Cliente e Servidor procede de forma implícita. Quando o Cliente espera a resposta da mensagem enviada para continuar o seu processamento, diz-se que o protocolo utilizado é um protocolo com bloqueio, onde o sincronismo entre Cliente e Servidor está implícito no mecanismo de passagem de mensagem. Caso o Cliente possa continuar suas tarefas, enquanto espera a resposta da mensagem, o protocolo de comunicação é um protocolo sem bloqueio [35].

Isto ocorre quando o sistema operacional do Cliente é do tipo multitarefa, possibilitando ao Cliente executar outras tarefas enquanto aguarda a resposta do Servidor. A teoria de programação concorrente é baseada na noção de processos de comunicação sendo executados em paralelo a outros processos. Esses processos se comunicam compartilhando memória ou passando mensagens por meio de um canal de comunicação compartilhado [01] [38]. O termo IPC se refere às técnicas utilizadas na passagem de mensagem.

No compartilhamento de memória, os processos concorrentes compartilham uma ou mais variáveis e utilizam a mudança de estados dessas variáveis para se comunicarem. Essa técnica inclui espera ocupada, semáforos, regiões críticas condicionais e monitores. Como esta técnica exige que os processos estejam na mesma máquina, não são considerados base para a programação Cliente/Servidor [35].

Em técnicas baseadas na passagem de mensagem, os processos enviam e recebem mensagens explicitamente, em vez de examinar o estado de uma variável compartilhada [01] [06]. O benefício principal da passagem de mensagem é que existe pouca diferença entre o envio de mensagens a processos remotos ou locais. Portanto, a passagem de mensagem é poderosa para criação de aplicações em rede. Outra vantagem é que mais informações podem ser trocadas numa mensagem do que por mudança no estado de uma variável compartilhada.

A Figura 3.7 mostra a troca de mensagem entre cliente e servidor sem o bloqueio por parte do cliente.

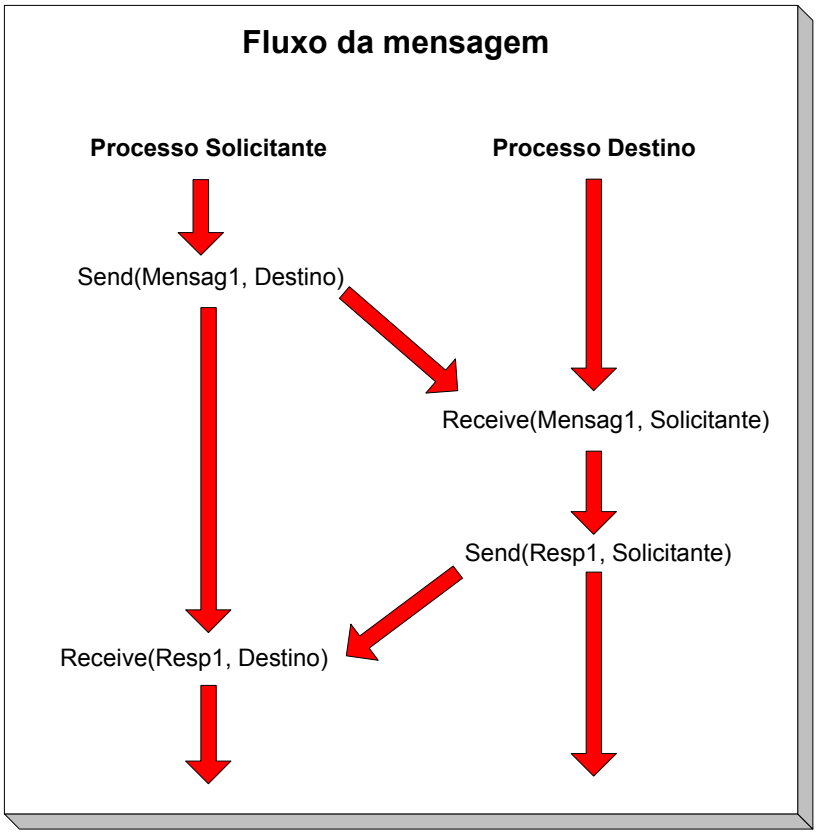

Figura 3.7 –Troca de mensagens entre processos origem e destino sem bloqueio

3.2.3.1 Características da conexão TCP

A técnica de estabelecimento de conexão do protocolo de transporte TCP é conhecida por *Three Way Handshake* ou aperto de mão triplo. Durante o estabelecimento de conexão são trocados três datagramas antes que se inicie, efetivamente, a troca de dados.

O primeiro datagrama é enviado com o bit *SYN* ativado e com o número de seqüência gerado através de um mecanismo baseado no uso de um relógio, com pulsos a cada 4 µs. A resposta deve vir com o bit *SYN* ativado, com um número de seqüência e com o bit *ACK* confirmando a recepção do primeiro datagrama. Por fim, o *host* que iniciou a conexão envia um último datagrama confirmando o recebimento da resposta.

As conexões TCP operam no modo *Full-Duplex*, ou seja, suportam transmissões de dados nos dois sentidos. Desta forma, independente de qual *host* tenha estabelecido a conexão, todos os dois podem enviar dados. Da mesma forma, a conexão poderá ser encerrada por qualquer um dos *hosts*.

Por se tratar de um protocolo mais complexo necessita de mais recursos para implementação deste tipo de conexão, deste modo são consumidos mais recursos para garantir o funcionamento de todas as etapas, entretanto com o avanço na tecnologia, que proporcionou o aumento na velocidade de processamento, este *overhead* diminuiu significativamente.

Outro detalhe importante é que devido a estas características e ao aumento do número de pacotes transmitidos, existe a necessidade de um controle do fluxo e também o controle de conexão. Na Figura 3.8 conforme [40], é apresentado um esquema através da linha do tempo para estabelecimento de uma conexão TCP.

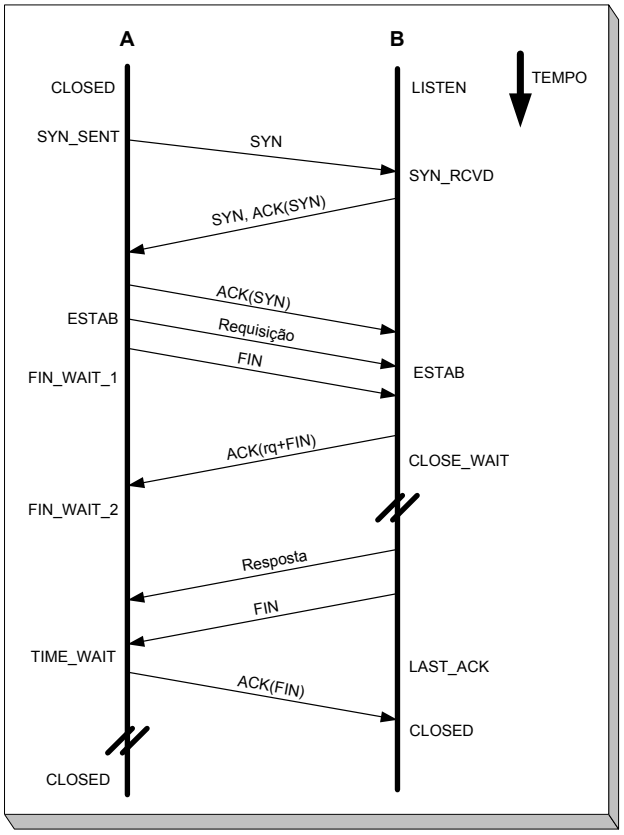

Figura 3.8 – Estabelecimento e encerramento de uma conexão TCP através da linha do tempo

Em conjunto com o protocolo IP, TCP/IP transformou-se no protocolo padrão utilizado pela Internet, que possibilita a interligação de diversas plataformas. Atualmente existem vários serviços que utilizam o protocolo TCP/IP, entre eles pode-se destacar alguns bem conhecidos como o FTP, *Telnet*, *E-mail*, *Ping*, *Finger* entre outros ilustrados através da Figura 3.9, que também apresenta a respectiva referência na camada do modelo OSI/ISO.

Na camada de Transporte pode-se notar a existência de dois padrões, o TCP, que é a comunicação com conexão e o UDP, que é a comunicação sem conexão. O padrão TCP permite a abertura de uma conexão virtual entre a máquina fonte e a destino em todas as camadas do modelo OSI/ISO.

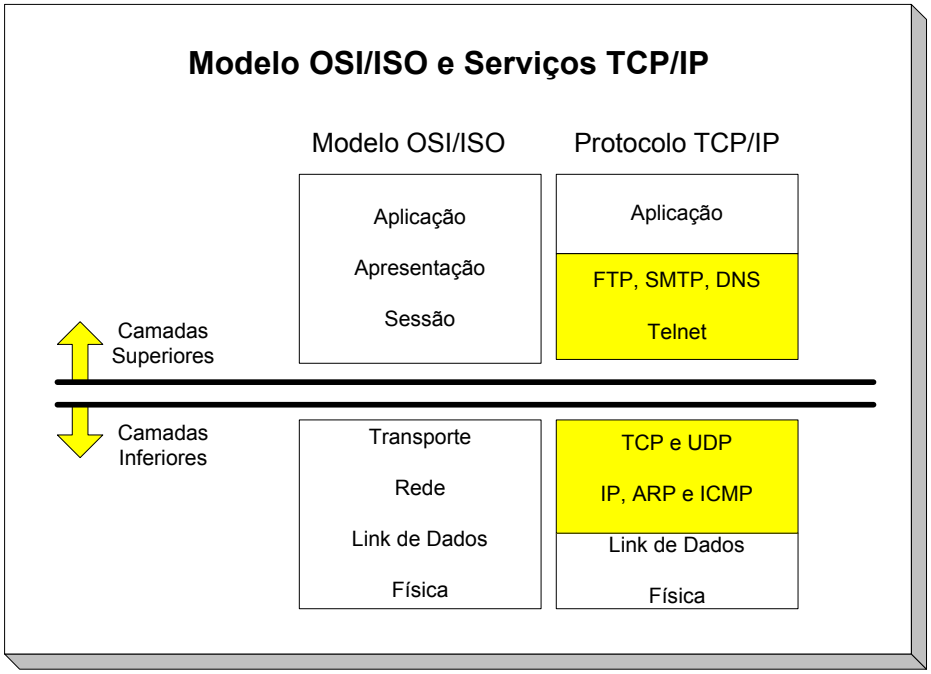

Figura 3.9 – Protocolo TCP/IP

A Internet utiliza o padrão UDP, uma vez que as comunicações baseadas no protocolo TCP/IP se utilizam de portas de comunicação que, associadas aos *sockets*, permitem a troca de mensagens. O protocolo TCP/IP disponibiliza 999999 portas (comprovação empírica), sendo que as portas de número 0 até a porta de número 1023 são reservadas para serviços pré-determinados, como por exemplo a porta 23, para o serviço *Telnet*, a porta 21 para o FTP e assim por diante. Devido a este limite, caso a Internet utilizasse o padrão TCP, as máquinas que respondessem a um número muito grande de acessos, acabariam limitando a sua utilização.

No padrão UDP, por sua vez, quando existe uma solicitação de comunicação, o endereço IP do remetente da mensagem segue junto com a mensagem para o destinatário, de forma que o disponibilizador do serviço possa posteriormente enviar a resposta do serviço solicitado.

É importante salientar que, para que as comunicações transcorram normalmente, as APIs não necessariamente precisam ser as mesmas, muito embora os protocolos devam ser os mesmos, como pode ser visto na Figura 3.10.

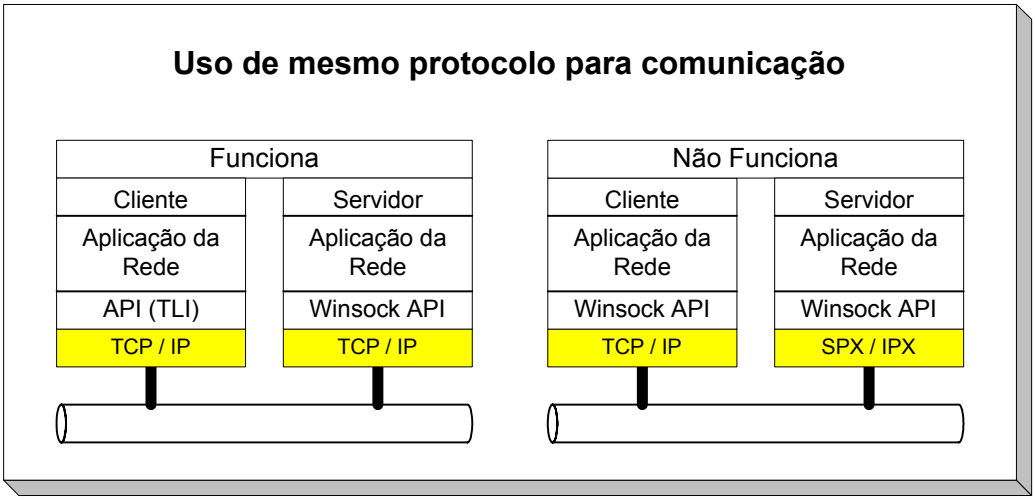

Figura 3.10 – Protocolos idênticos para que haja a comunicação

Os demais protocolos suportados pelo *Windows NT* permitem a conexão com outros tipos de plataformas, como pode ser visto na Tabela 3.3.

Protocolo	**Plataforma**
NetBEUI	*Microsoft*
SPX/IPX	*Novell*
Apple Talk	*Macintosh*

Tabela 3.3 – Exemplos de Protocolos do *Windows* NT/2000

3.2.3.2 NETBEUI, SPX/IPX e Appletalk

O protocolo NetBEUI é de propriedade da *Microsoft* e permite a conexão com máquinas que estejam com os ambientes *Windows 3.x, Windows NT 3.x / 4.0* e *Windows 95* e posteriores [10]. Ele é o responsável pela camada de Transporte e Rede, acrescentando uma extensão para a camada de *Link* de Dados. Conecta-se diretamente com o NetBIOS na camada de sessão [28], que oferece as funcionalidades de transporte e rede do NetBEUI. Possui dois tipos de *frames*, um para fornecer um fluxo de dados confiável e o outro para *frames* de informação não numerados usados em datagramas.

Os protocolos SPX/IPX são uma implementação dos protocolos XNS da *Xerox*. O protocolo SPX, igual ao SPP da *Xerox*, oferece um serviço de fluxo de dados confiável, enquanto que o IPX, igual ao IDP, oferece um serviço de datagrama, permitindo a conexão entre máquinas do tipo PC que estejam rodando o Sistema Operacional de Rede da *Novell* [28].

Por sua vez, o protocolo *Apple Talk* permite a conexão com máquinas da plataforma *Macintosh* e é um protocolo totalmente particular. Os principais protocolos usados por máquinas Macintosh são os DDP - *Datagram Delivery Protocol*, ATP – *AppleTalk Transaction Protocol* e ADSP - *AppleTalk Data Stream Protocol* [28].

3.2.3.3 *Sockets*

O *Socket* teve origem na Universidade de *Berkeley*, como sendo a API de desenvolvimento do protocolo TCP/IP para o ambiente UNIX [06]. Ele é um dos mecanismos que o IPC possui para a troca de mensagens entre processos.

Um *Socket* é similar a um descritor de arquivo. Ele identifica o ponto final (*endpoint*) para a comunicação e é implementado como um inteiro positivo [30]. Existe uma diferença sutil, porém importante, entre descritores de arquivos e *socket*.

Um descritor de arquivo é ligado a um arquivo ou dispositivo específico quando criado pelo comando *open*. Um descritor de *socket* não é ligado a local algum quando criado pelo comando *socket*. Uma aplicação pode decidir ligar-se a um endereço explicitamente usando um comando *bind* ou pode fornecer endereços dinamicamente quando envia datagramas usando o comando *send*. Portanto, *sockets* podem ser usados como uma interface para transportes de rede baseados em conexão e sem conexão [06] [11].

O *socket*, que é um manipulador (*handle*), está associado a um largo conjunto de dados armazenados na implementação do protocolo de rede. O termo *socket* é usado no UNIX para fornecer uma interface tipo arquivo às redes.

Quando uma operação para criar um *socket* é chamada, o sistema retorna um manipulador, como descritor de *socket*. Esse descritor é usado em todos os outros comandos relacionados ao *socket* e é intercambiável com o descritor de arquivo usado em funções *read* e *write*.

Os dados associados ao *socket* incluem várias informações, como o endereço IP das máquinas que estão se conectando, as portas dos dois lados da conexão TCP e o estado da conexão corrente.

A camada de *sockets* dentro do contexto da comunicação corresponde a camada de Sessão do Modelo OSI/ISO, que tem como função o gerenciamento das sessões de comunicação processo a processo entre os *hosts*. Ela é também responsável por estabelecer e encerrar as conexões entre os aplicativos cooperantes [11].

Dentro do protocolo TCP/IP, a camada TCP abrange a camada de Sessão e Transporte do Modelo OSI/ISO, favorecendo diretamente a programação dos *sockets* sobre o protocolo TCP/IP.

Para o ambiente *Windows* foi desenvolvido o *Windows Sockets* - *Winsock*, visando facilitar a interconexão entre as estações *Windows*, através do protocolo TCP/IP, que é base para o acesso a Internet. *Windows Sockets* é uma interface aberta para programação em rede sob o *Microsoft Windows* [27]. A especificação da interface do *Windows sockets* define claramente a divisão entre a aplicação de rede e o protocolo da rede, sua versão mais recente é o *winsock32* 5.0 (revisão 2.6) para *Windows* 2000/XP e para *winsock* é a versão 3.10. A partir versão 2.0 da especificação do *Windows sockets* se associa a arquitetura que o *Windows NT* utiliza para suportar múltiplos protocolos de fornecedores variados [27].

O *Winsock* API aumenta a funcionalidade dos *sockets* de *Berkeley*, ao acrescentar extensões específicas do *Windows* para suportar a natureza orientada a mensagens do S.O. baseado no *Windows*. Todas as aplicações que hoje em dia acessam a Internet diretamente da residência do usuário, usando FTP, *E-mail, Finger, Telnet*, SMTP, entre outros, utilizam os *sockets* como base de comunicação, através do protocolo TCP/IP [30].

Os Sistemas Operacionais *Windows NT/2000* e *Windows* 95 em diante já possuem dentro de suas bibliotecas, as rotinas para suportar a programação para o protocolo TCP/IP via *sockets* [10]. Para o ambiente *Windows*, o relacionamento entre as aplicações e o ambiente de rede, pode ser demonstrado na Figura 3.11.

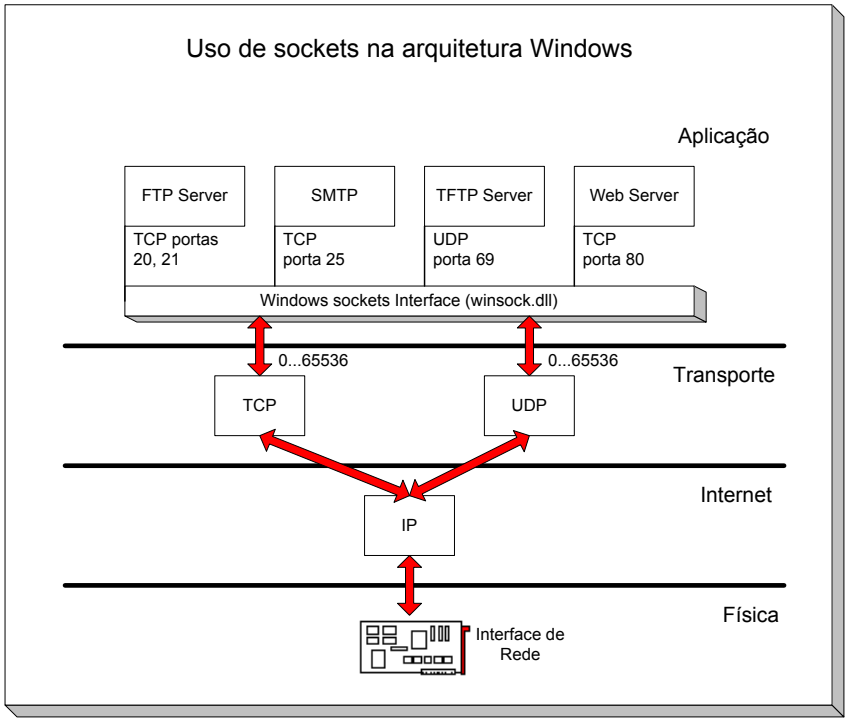

Figura 3.11 – Relacionamento da biblioteca *sockets* em ambiente *Windows*

Na API do *sockets* é possível determinar se a comunicação que se vai estabelecer é com conexão - *STREAM*, ou sem conexão - *DATAGRAM*. Para se trabalhar numa comunicação baseada em conexão, na camada de transporte do TCP/IP, o protocolo utilizado é o TCP, determinando que somente após o estabelecimento de uma comunicação é possível efetuar troca de mensagens.

Para a abertura de uma conexão com *sockets*, é necessário que várias funções da biblioteca *Winsock* sejam chamadas. A Figura 3.12 demonstra a seqüência de operações tanto no lado Servidor quanto no lado Cliente.

No caso de se estabelecer uma conexão *Stream*, o Servidor é primeiramente inicializado. A função *Socket()* define o descritor no qual a aplicação se associará quando desejar transmitir uma mensagem. Posteriormente a função *Bind()* interliga o descritor ao endereço IP da máquina servidora e a porta TCP/IP na qual as transmissões irão ocorrer. A função *Listen()* permite ao Servidor ficar "escutando" qualquer solicitação de conexão.

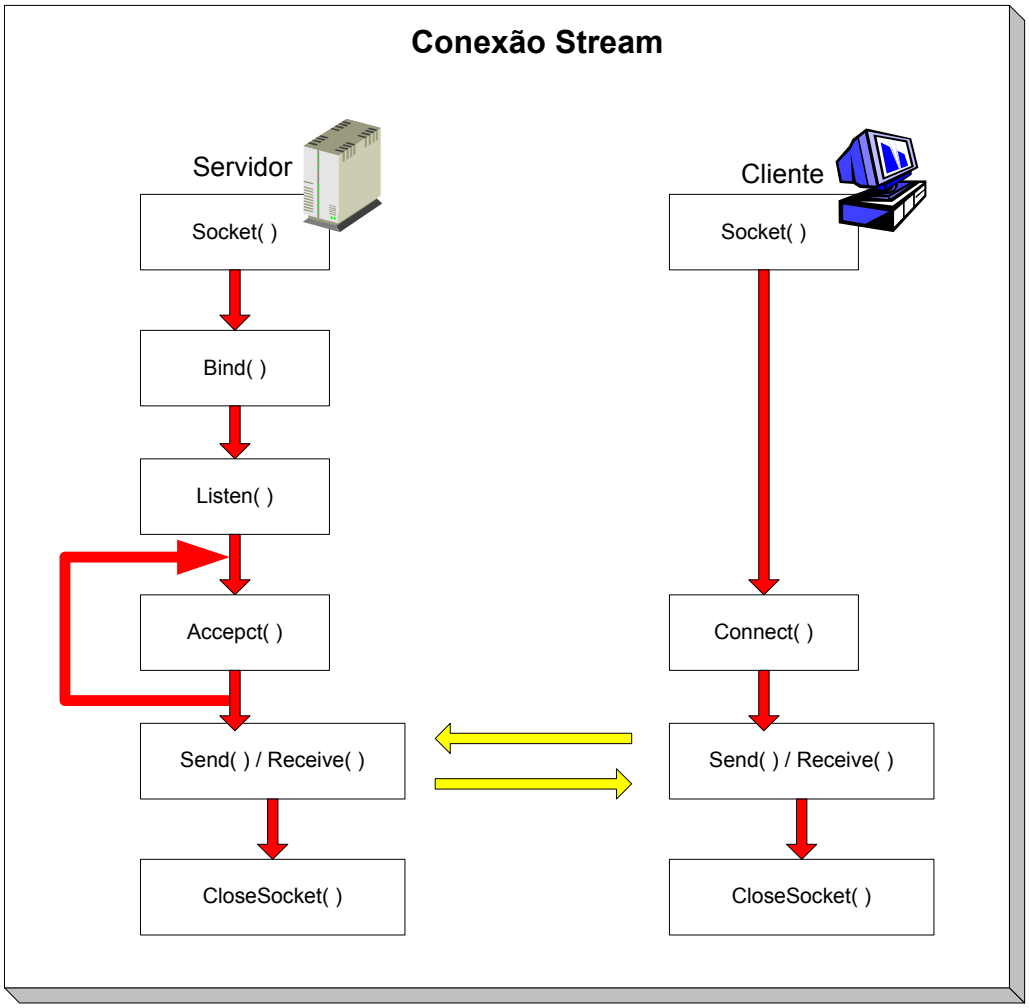

Figura 3.12 – Conexão Bitstream

Quando uma solicitação chega de um Cliente após ativar a função *Connect()*, o Servidor cria um processo filho mediante a função *Accept()*, numa nova porta TCP/IP. O pedido do Cliente é associado a esta porta, permitindo assim a transferência de dados pelas funções *Send()/Recv()*, deixando, desta forma, a porta original de conexão do Servidor livre para efetuar novas conexões.

Ao término da comunicação, o Cliente utiliza a função *CloseSocket()* para fechar a conexão, liberando a porta do processo filho do Servidor para ser ligada a outro processo de comunicação. O Servidor só irá utilizar esta função quando for desligado.

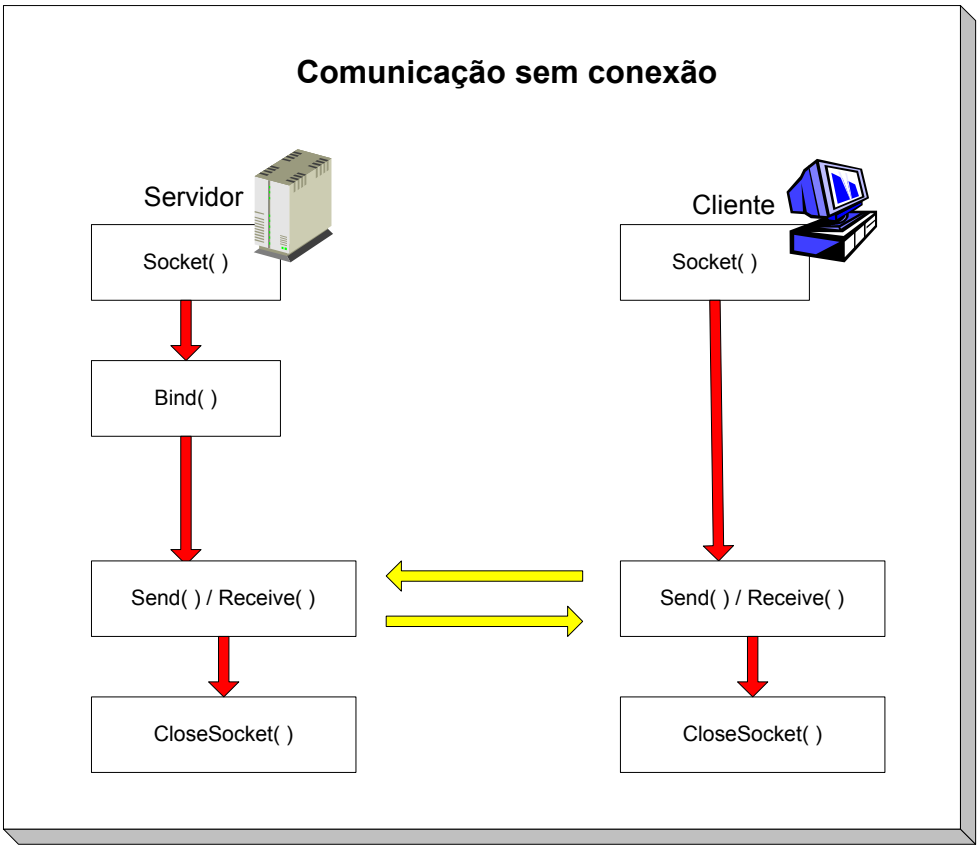

Figura 3.13 – Conexão datagrama

Por sua vez, para uma comunicação sem conexão, conforme a Figura 3.13, a utilização das funções de *sockets* tornam-se mais fáceis de se implementar, porém, é necessário lembrar que a utilização de comunicação baseada em Datagramas determina que as rotinas de recuperação das seqüências dos pacotes entregues a rede devem ser programadas, aumentando assim a possibilidade de falhas na implantação do sistema.

A seqüência e as funções definidas para uma conexão Datagrama são as mesmas da conexão *Stream*, excluindo apenas as funções *Listen(), Accept()* e *Connect()*. Isto ocorre pelo fato do protocolo baseado em Datagrama não utilizar uma conexão real entre as máquinas origem e destino, implementando, automaticamente dentro das funções de troca de mensagens, *Send()/Recv()*, um cabeçalho com o endereço IP da máquina no qual será feita a comunicação.

Um ponto a ser observado, é que, para a transmissão, tanto na função *Send()* ou *Recv()*, os *sockets* somente operam com dados do tipo *Char*. Para a transmissão de dados puramente do tipo caracter, o sistema desenvolvido baseado em *sockets* não impõe nenhuma restrição, entretanto, para sistemas

desenvolvidos sob o paradigma da Orientação a Objeto, que manipula tanto objetos simples, como complexos ou longos, é necessário que, ao se transmitir, se faça uma conversão da classe do objeto para o tipo *Char*. No momento em que a informação chega ao seu destino, é necessário que o receptor faça a reconversão, isto é, do tipo *Char* para o tipo original do objeto.

Esta peculiaridade para a transmissão de dados de tipos diferentes de *Char*, faz com que o destinatário da mensagem possua o conhecimento prévio de todos os tipos possíveis de objetos que podem ser transmitidos.

3.2.4 Princípios da comunicação

Quando há a comunicação entre duas entidades distintas, é importante levar em consideração alguns aspectos da comunicação que precisam ser aplicados e que serão mostrados mais adiante. No caso da computação existem ferramentas suficientes para prover estes princípios e um ramo de pesquisa em plena evolução, a criptografia computacional.

Não é objetivo deste trabalho aprofundar-se em temas como sistemas criptográficos, pois trata-se de um assunto complexo que necessitaria muito mais que um capítulo para apresentá-lo. Mesmo assim é de fundamental importância citar os princípios que regem uma comunicação, pois considerando que o meio de comunicação é vulnerável, as entidades envolvidas na troca de informações devem garantir a segurança e o sigilo, bem como estarem preparadas contra atacantes que conhecem e exploraram tais vulnerabilidades.

3.2.4.1 Princípio da Disponibilidade

Uma grande variedade de ataques pode resultar na perda ou redução da disponibilidade da informação. Alguns desses ataques são compensados através de medidas automatizadas, como a autenticação e a criptografia, ao passo que já outros requerem algum tipo de ação física para a prevenção ou recuperação das perdas de disponibilidade de elementos de um sistema distribuído [18] [39].

3.2.4.2 Princípio da Integridade

O serviço de integridade pode ser aplicado a todo um fluxo de mensagens de uma conexão, a uma única mensagem ou a determinados campos desta mensagem. Uma conexão que tenha este princípio implantado garante que as mensagens serão recebidas como foram enviadas, sem duplicação, inserção indevida, modificações, sem reordenação ou repetições. A destruição de dados também é tratada neste

serviço. Sob outro foco, este serviço trata tanto da modificação da mensagem como da negação de serviços.

É possível fazer uma distinção entre o serviço com e sem recuperação. Pois o serviço de integridade trata de ataques ativos, a atenção se concentra na detecção ao invés da prevenção [18] [39].

Caso uma violação de integridade seja detectada, então o serviço pode simplesmente informar esta violação, de forma que uma outra parte do software ou algum tipo de intervenção humana seja necessário para a recuperação de tal violação. De forma alternativa, existem mecanismos disponíveis para a recuperação de perda de integridade de dados. Esta última alternativa é a mais atraente.

3.2.4.3 Princípio da Confidencialidade

A confidencialidade é a proteção das informações contra ataques passivos e análise de mensagens, quando em trânsito nas redes ou contra a divulgação indevida da informação, quando sob guarda [18] [39].

Com respeito a utilização indevida de conteúdos de mensagens, pode-se identificar diversos níveis de proteção para cada tipo de informação identificado. Podem ser definidos diversas formas para este serviços, incluindo a proteção de mensagens individuais ou até mesmo de campos dentro desta mensagem. Este processo de identificação e refinamento daquilo que realmente deve ser protegido é bastante complexo e se reflete em toda a estrutura de segurança adotada.

Como serviços derivados dos princípios de segurança anteriormente citados, podemos identificar:

3.2.4.4 Princípio da Autenticidade

O serviço de autenticação se relaciona com a garantia de que a comunicação é autentica. No caso de uma simples mensagem, como é o caso de um sinal de alarme, a função da autenticação é garantir ao receptor que a mensagem é realmente originária da fonte informada [18] [39].

No caso de uma interação em tempo real, como a conexão de um computador com outro, pode-se considerar dois aspectos, o primeiro no momento da inicialização da conexão, este serviço deve garantir que as duas entidades são autenticas, ou seja que são quem alegam ser. Em segundo lugar, o serviço deve garantir que a comunicação deve ocorrer de forma que não seja possível a uma terceira parte se disfarçar e se passar por uma das partes já autenticadas na inicialização da conexão para conseguir transmitir e receber mensagens de forma autorizada.

3.2.4.5 Não Repúdio

Este serviço previne tanto o emissor contra o receptor, quanto previne contra a negação de uma mensagem transmitida [18] [39]. Desta forma, quando uma mensagem é enviada, o receptor pode provar que de fato a mensagem foi enviada pelo emissor em questão. De forma similar, quando uma mensagem é recebida, o emissor pode provar que a mensagem foi realmente recebida pelo receptor em questão.

3.2.4.6 Controle de Acesso

No contexto de segurança de rede, o controle de acesso é a habilidade de limitar ou controlar o acesso aos computadores hospedeiros ou aplicações através dos enlaces de comunicação e do controle de acesso físico [18] [39]. Para tal, cada entidade que precisa obter acesso ao recurso, deve primeiramente ser identificada, ou autenticada e de forma a que os direitos e permissões de acesso sejam atribuídos ao usuário.

Conclusão

As aplicações estão evoluindo e criando novas tendências para atender a demanda de mercado, porém o conhecimento do ambiente de comunicação e suas particularidades são fatores fundamentais para o sucesso e sobrevivência destas aplicações.

Com o crescimento da Internet interligando computadores através da rede e o estabelecimento do protocolo TCP/IP como padrão de *facto*, há uma tendência para que os protocolos proprietários caiam em desuso.Algumas empresas resolveram este problema utilizando seus protocolos proprietários encapsulados dentro de pacotes TCP/IP, garantindo assim que aplicações pudessem ser migradas para esta nova realidade.

Capítulo 4

Provendo segurança nos protocolos de comunicação

O propósito deste Capítulo é mostrar que apesar das fraquezas existentes nos protocolos de comunicação utilizados nas aplicações cliente/servidor, existem mecanismos que aliados a outras técnicas, como por exemplo, a distribuição física da rede (estações, servidores locais, servidores públicos) e *firewalls* possibilitam agregar segurança ao meio.

Protocolos para estabelecimento de canais seguros e métodos para garantir a autenticação vinda de uma rede externa ou pública, podem prover a solução para o problema da segurança das informações e dos usuários que acessam os dados.

No decorrer deste Capítulo haverá uma breve exposição dos principais protocolos seguros e também dos métodos de autenticação.

Não aprofundaremos no funcionamento detalhado destes para que não desviemos do foco principal deste trabalho e incorrermos no erro de estender o documento apresentando uma série de características que estão mais ligadas a aspectos criptográficos, apenas este tópico pode gerar uma nova dissertação.

4.1 Protocolos seguros para Aplicações

4.1.1 PPP

Point-to-Point é um protocolo de acesso remoto utilizado pelo PPTP para enviar dados multi-protocolos através de uma rede baseada em TCP/IP. Pode encapsular pacotes IP, IPX e NetBEUI entre quadros PPP e enviar os pacotes encapsulados gerando um enlace ponto-a-ponto entre os computadores que enviam e recebem a informação. O protocolo PPP é utilizado para criar uma conexão do tipo discada entre o cliente e o servidor de acesso e para tal executa as seguintes tarefas:

- **Estabelece e finaliza a conexão física**. O protocolo PPP utiliza uma seqüência definida na RFC 1661 para estabelecer e manter conexões entre computadores remotos.

- **Autentica usuários**. Clientes PPTP são autenticados utilizando-se o protocolo PPP. Texto claro, criptografado ou no caso da *Microsoft* (estações *windows*) *Microsoft encrypted authentication* podem ser usados pelo protocolo PPP.

- **Gerar datagramas PPP que contem pacotes IPX, NetBEUI ou TCP/IP criptografados**. PPP gera datagramas que contém um ou mais dados criptografados dos pacotes IPX, NetBEUI ou TCP/IP.

Devido ao fato dos pacotes da rede serem criptografados, o tráfego entre o cliente PPP e o servidor de acesso é um canal seguro. A Figura 4.1 ilustra todo o processo.

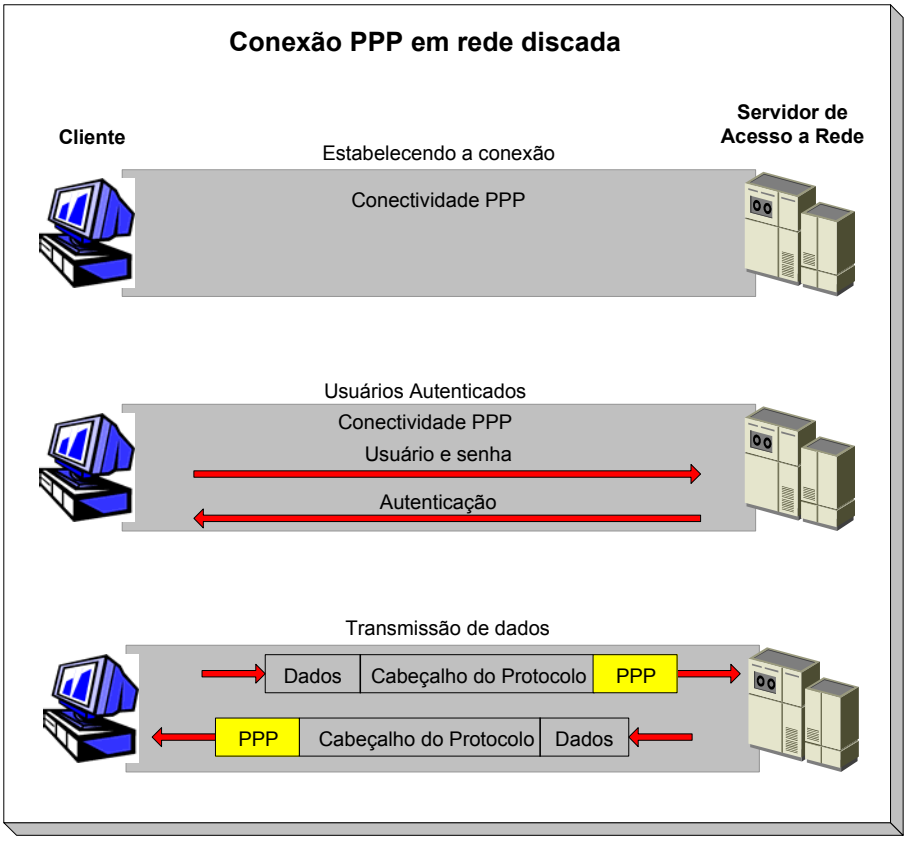

Figura 4.1 – Etapas da conexão do protocolo PPP

Uma observação importante a fazer sobre este protocolo é o fato de em muitas situações, clientes remotos possuem um acesso direto a uma rede TCP/IP, não necessariamente o acesso provêm de uma rede discada.

4.1.2 O protocolo PPTP

Point-to-Point Tunneling Protocol é um protocolo de rede que possibilita a transferência de dados de forma segura, partindo de um cliente remoto para um servidor que se encontra numa rede local privada criando um *virtual private network* (VPN) através de redes de dados baseadas em TCP/IP. PPTP oferece suporte para multi-protocolos e VPNs através de redes públicas como a Internet.

A tecnologia de rede do PPTP é uma extensão de acesso remoto do protocolo *Point-to-Point* (PPP) definido em um documento da *Internet Engineering Task Force* (IETF) intitulado "*The Point-to-Point Protocol for the Transmission of Multi-Protocol Datagrams over Point-to-Point Links*" originando a RFC 1171 [21].

PPTP é o protocolo de rede que encapsula pacotes PPP dentro de datagramas IP para transmissão sobre a Internet ou outras redes públicas baseadas em TCP/IP. Pode apenas ser utilizado para interligação em redes privadas locais, ou seja *LAN-to-LAN*. Foi desenvolvido para prover um método seguro para interligar redes privadas sobre a Internet e, examinando sua construção, observa-se que a comunicação segura se faz através de três processos, cada qual necessita da execução com sucesso do anterior. São eles:

- **PPP Comunicação e Conexão**. Um cliente PPTP utiliza o protocolo PPP para se conectar a um ISP[1], geralmente através de uma linha telefônica ou ISDN[2]. Esta conexão utiliza o protocolo PPP para estabelecer a conexão e criptografar os dados no pacote.
- **PPP Controle de Conexão**. Usando uma conexão Internet estabelecida pelo protocolo PPP, o protocolo PPTP gera um controle de conexão do cliente PPTP para um servidor PPTP na Internet. Esta conexão utiliza o transporte TCP para estabelecer a conexão que é denominada de túnel PPTP.
- **PPTP Tunelamento de dados**. Na última etapa o protocolo PPTP gera datagramas contendo pacotes criptografados PPP que são então enviados através do túnel PPTP ao servidor PPTP. Ao receber os datagramas, estes são desmontados e decriptografados novamente tornando-se pacotes PPP que finalmente são encaminhados para a rede local.

4.1.2.1 Controle de Conexão PPTP

O protocolo PPTP especifica uma série de mensagens de controle enviadas entre o cliente e o servidor PPTP. Este controle estabelece, mantém e encerra o túnel PPTP. A lista de mensagens utilizadas é relativamente simples. Na Tabela 4.1 são apresentadas as mensagens para estabelecer e manter um túnel PPTP.

Mensagem	Finalidade
PPTP_START_SESSION_REQUEST	Inicia uma sessão
PPTP_START_SESSION_REPLY	Replica a requisição para iniciar sessão
PPTP_ECHO_REQUEST	Mantém a sessão
PPTP_ECHO_REPLY	Replica o pedido de manutenção da sessão
PPTP_WAN_ERROR_NOTIFY	Notifica um erro que ocorre na conexão PPP
PPTP_SET_LINK_INFO	Configura a conexão entre o cliente e o servidor PPTP

[1] ISP: *Internet Service Provider*
[2] ISDN: *Integrated Services Digital Network*

Mensagem	Finalidade
PPTP_STOP_SESSION_REQUEST	Finaliza a sessão
PPTP_STOP_SESSION_REPLY	Replica o pedido de finalização da sessão

Tabela 4.1 – Mensagens do protocolo PPTP

Mensagens de controle são transmitidas em pacotes de controle em um datagrama TCP. Uma conexão TCP é criada entre o cliente PPTP e o servidor PPTP. Esta conexão é utilizada para troca de mensagens de controle. Um datagrama contém um cabeçalho PPP, um cabeçalho TCP, um controle de mensagens PPTP e os *traillers* apropriados, como pode ser visto na Figura 4.2 [21].

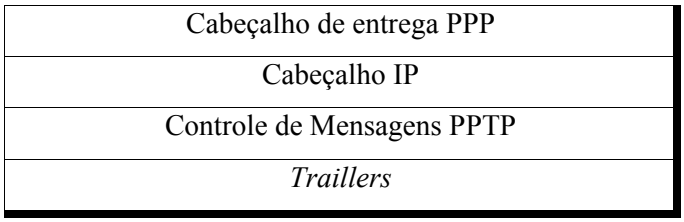

Figura 4.2 – PPTP Datagrama TCP com mensagens de controle

A troca de mensagens entre o cliente PPTP e o servidor sobre conexões TCP são utilizadas para criar e manter um túnel PPTP. Este processo é ilustrado na Figura 4.3.

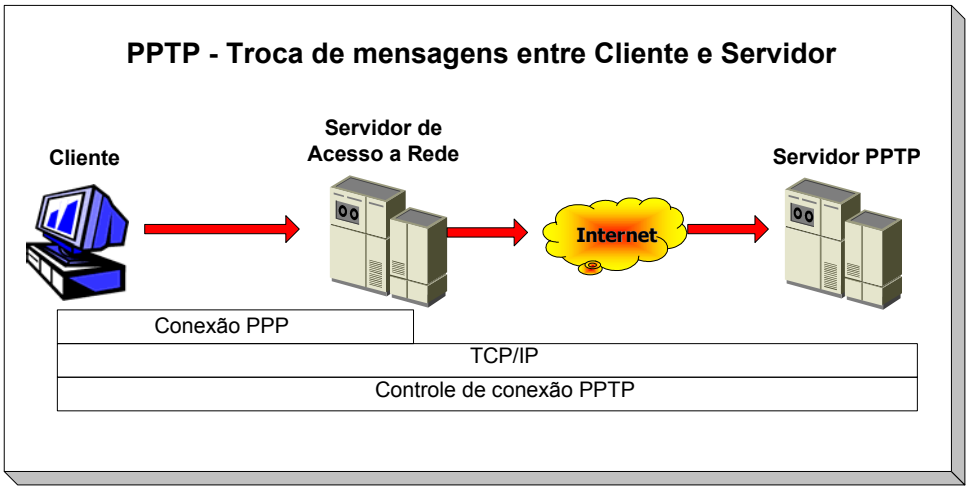

Figura 4.3 – PPTP troca de mensagens entre Cliente e Servidor

4.1.2.2 Transmissão de dados no protocolo PPTP

Após o estabelecimento do túnel PPTP, os dados do usuário são transmitidos entre o cliente e o servidor PPTP. Os dados são transmitidos em um datagrama IP contendo pacotes PPP. O datagrama IP

é gerado utilizando-se uma versão modificada do protocolo *Generic Routing Encapsulation* (GRE), definido na RFC 1701 e 1702. O datagrama IP criado pelo protocolo PPTP é simular ao apresentado na Figura 4.4.

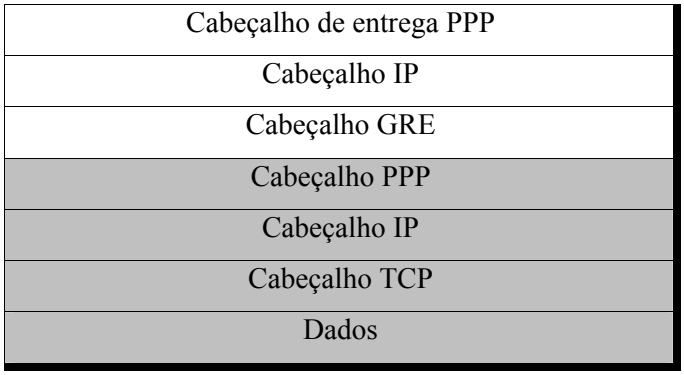

Figura 4.4 – Datagrama IP contendo pacote PPP gerado pelo protocolo PPTP

O cabeçalho de entrega PPP provê a informação necessária para que o datagrama seja trafegado pela Internet, o cabeçalho GRE é utilizado para encapsular o pacote PPP com o datagrama IP. O pacote PPP foi gerado pelo Servidor de Acesso a Rede (ISP).

Nos servidores *Microsoft* como *Windows NT/2000/2003 Server* este serviço é denominado de RAS[1]. É importante salientar que o pacote PPP é um bloco ilegível devido a criptografia aplicada, caso a conexão seja capturada os dados estão seguros.

4.1.3 SSL

O *Secure Socket Layer* (SSL, atualmente na versão 3) é um protocolo de comunicação que implementa um duto seguro para comunicação de aplicações na Internet, de forma transparente e independente da plataforma. Foi desenvolvido pela *Netscape Communications* em sua versão inicial em julho de 1994.

Em abril de 1995 foi lançada a referência para implementação da versão 2 (sendo distribuído junto os *Browsers Netscape* e *Internet Explorer* e os servidores *web* mais comuns - Apache, NCSA httpd, IIS[2], *Netscape Server* dentre outros), apoiado pela *Verisign* e *Sun*, transformando-se em um padrão em *e-commerce*, tendo a sua especificação submetida ao grupo de trabalho W3C[3] (especificação disponível em um "*draft*" do IETF com o nome de TLS - *Transaction Layer Security*) [14].

[1] RAS: *Remote Access Service*
[2] IIS: *Internet Information Service*
[3] W3C: *World Wide Web Consortium*

Em novembro de 1995 foi lançada a versão 3 do SSL, tendo como melhorias a diminuição no número de rodadas de negociação, a escolha das cifras e compressão por parte do servidor, um suporte mais completo para a troca de chaves de algoritmos de cifragem, a possibilidade de renegociação das cifras em uso e a separação das chaves de autenticação e criptografia.

Embora as diferenças entre o TLSv1 e o SSLv3 não sejam grandes, são suficientes para que eles não possam interoperar diretamente. Caso seja necessário, o TLSv1 pode emular o SSLv3 [14]. A proposta do SSL é permitir a autenticação de servidores, criptografia de dados, integridade de mensagens e, como opção, a autenticação do cliente, operando nas comunicações entre aplicativos de forma interoperável. Visa garantir os seguintes objetivos:

- **Segurança criptográfica** para o estabelecimento de uma ligação segura entre duas máquinas/aplicativos, assegurando a privacidade na conexão, com a utilização de algoritmos simétricos (como o DES ou RC4) que negociam uma chave secreta na primeira fase do *handshaking* (usando chaves públicas – assimétricas);
- **Autenticação do Servidor** (opcionalmente também do Cliente) por meio de algoritmos assimétricos como o RSA ou o DSS[1];
- **Confiabilidade na conexão**, conseguida com o uso de Códigos de Autenticação de Mensagens (MAC).

O SSL também permite a montagem de um *framework* onde outras chaves públicas e métodos de criptografia podem ser utilizados, evitando a necessidade de implementação de toda uma pilha de protocolos (com os riscos da introdução de fraquezas). Como uma vantagem adicional, a questão do desempenho foi levada em consideração no projeto para reduzir o número de conexões e minimizar o tráfego na rede, opcionalmente pode ser usado um esquema de *cache* em memória durante o estabelecimento da sessão, com a finalidade de reduzir o número de conexões e reduzir a atividade no acesso a rede.

O SSL atua entre as camadas transporte (TCP) e aplicação, sendo independente do protocolo de alto nível podendo ser executado sob *HTTP, Telnet, FTP, SMTP* e outros, de forma transparente [14]. Ele implementa duas novas camadas, sobre o TCP/IP, conforme apresentado na Figura 4.5.

[1] DSS: *Digital Signature Standard*

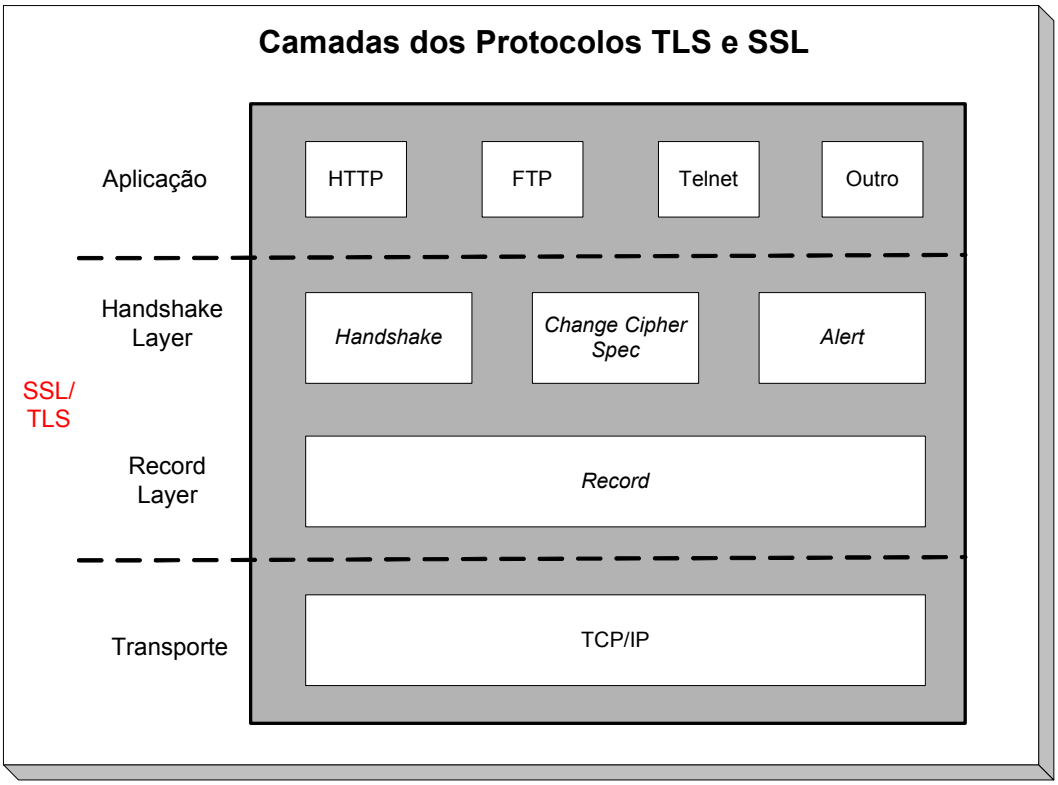

Figura 4.5 – Camadas Implementadas pelos protocolos SSL e TLS

SSL Handshake Protocol: Faz a autenticação entre cliente e servidor, cuidando da inicialização da comunicação, permitindo a negociação do algoritmo de criptografia e as chaves criptográficas iniciais. Utiliza as chaves assimétricas para fazer a negociação inicial, abrindo um canal seguro para o envio da chave simétrica de sessão, criada de forma aleatória. Opera sobre o *Record Layer*. Todas as mensagens da negociação utilizam o MAC e funções de *hash* (como SHA, MD5 e outras) para aumentar a segurança do processo inicial. A ordem das mensagens é absoluta e seus conteúdos são manuseados pela *Record Layer*. Ao estabelecer a conexão, o *Handshake Protocol* estabeleceu o identificador de sessão, o conjunto criptográfico (*cypher suite*) a ser adotado e o método de compressão a ser utilizado. O conjunto criptográfico negociado define três algoritmos:

- Um algoritmo para troca de chaves;
- Um algoritmo para cifragem de dados, e
- Um algoritmo para inserção de redundância nas mensagens.

SSL Change Cipher SPEC Protocol: Sinaliza as transições nas estratégias de cifragem. Constitui-se de uma única mensagem que pode ser transmitida tanto pelo cliente como pelo servidor para notificar que os próximos blocos utilizarão chaves criptográficas recém negociadas.

SSL Alert Protocol: Acompanham os erros na *Record Layer*, fazendo troca de mensagens para sinalizar problemas com a seqüência de mensagens, erros de certificação ou criptografia.

SSL Record Layer Protocol: Encapsula as camadas de nível mais alto (quando conjugado com o HTTP, implementa o HTTPS), provendo os seguintes serviços:

Fragmentação (transforma blocos de dados em registros *SSLPlaintext* de, pelo menos, 224 bytes)

Compressão, (transforma os registros *SSLPlaintext* em registros *SSLCompressed*, utilizando os algoritmos negociados no *handshake*).

Autenticação de mensagem, acréscimo do MAC e número seqüencial (antes da criptografia).

Criptografia (as funções definidas no *handshake* são definidas na mensagem *SSLCipherSpec* e são utilizadas para transformar o *SSLCompressed* em *SSLCiphertext*). A comunicação é iniciada pelo estabelecimento de uma sessão, caracterizada pelo **Estado da Sessão** e o **Estado da Conexão**.

O Estado de Sessão é constituído pelos seguintes elementos:

- ***Session Identifier***: Uma seqüência arbitrária de bytes escolhida pelo servidor para identificar a sessão.
- ***Peer certificate***: Certificado do *peer* (opcionalmente pode ser nulo).
- ***Compressiom Metod***: Algoritmo utilizado na compressão.
- ***Cipher Spec***: Especifica o algoritmo usado na criptografia (*null*, DES entre outros) e um algoritmo MAC (MD5 ou SHA).
- ***Master Secret***: Uma chave secreta de 48 *bytes* trocada entre cliente e servidor.
- ***Is Resumable***: *Flag* que indica se a sessão pode ser utilizada em outras conexões.

O Estado de Conexão é constituído pelos seguintes elementos:

- ***Server and Client Random***: Seqüência de bytes aleatórios escolhidos pelo servidor e cliente a cada conexão.
- ***MAC Secret***: Segredo usado nas operações MAC na escrita de dados.
- ***Write Key***: Chave de criptografia usada pelo cliente e servidor para criptografar e descriptografar.

- ***Inicialization Vetors***: Utilizados no algoritmo de criptografia.
- ***Sequence Numbers***: Utilizados no algoritmo de criptografia.

Na Tabela 4.2 temos um resumo dos algoritmos disponíveis utilizados no SSL versão 3.

Algoritmos	**Utilização**
NULL, RSA, *Diffie-Hellman RSA*, *Diffie-Hellman DSS*, *DHE_DSS*, *DHE_RSA*, *DH_anonymous, Fortezza/DMS*	Troca de chaves de sessão, durante o *handshake*
NULL, RC2, RC4, IDEA, DES, 3DES, Fortezza	Definição de chave de criptografia
NULL , SHA , MD5	Implementação da função de *Hash* para definição do MAC
X.509 v1, X.509 v2, X.509 v3	Certificados

Tabela 4.2 – Descrição dos algoritmos disponíveis utilizados pelo SSL

O sucesso do SSL se deve ao fato de ser um dos protocolos mais convenientes e utilizados para implementação de transações seguras. Sua implementação é relativamente simples, colocando-se o SSL no topo da pilha TCP/IP e substituindo as chamadas TCP pelas chamadas SSL [14]. Trabalha independente das aplicações utilizadas e, após o *handshake* inicial, comporta-se como um canal seguro que permite que se execute todas as funções que normalmente estão disponíveis no TCP/IP.

Outro fator importante é a disponibilidade de primitivas necessárias para conexões seguras, a saber: autenticação, troca de chaves de sessão com o uso de criptografia assimétrica prévia, criptografia com métodos simétricos, MAC e certificação. O IETF[1] está trabalhando na sua padronização formal, denominada TLS, que será apresentada na seção 4.1.4.

4.1.4 TLS

O protocolo TLS (*Transport Layer Security*), descrito na RFC 2246, tem por objetivo prover privacidade e integridade de dados entre duas aplicações que utilizam redes públicas/abertas como meio para comunicação. É o padrão do IETF e o sucessor do *Netscape SSL,* sua construção foi baseada no *SSL v3,* no fundo o TLS não difere muito do SSL v3, eles não são compatíveis, mas em termos de

[1] IETF: *Internet Engineering Task Force*

segurança, não há nenhuma diferença. O TLS nada mais é do que uma padronização do SSL v3. Hoje o TLS é suportado pela maioria dos navegadores, por exemplo, o Internet Explorer 5.0 [25].

É um protocolo independente da aplicação, ou seja, outros níveis de protocolo podem estar acima do TLS de forma transparente. As decisões sobre como iniciar o TLS e como interpretar os certificados de autenticação trocados entre as partes da comunicação, ficam a cargo da aplicação que executa acima do TLS.

Permite que aplicações cliente-servidor comuniquem-se prevenindo: escuta de conversas privadas, modificação das mensagens sem permissão e cópia ilegal de mensagens [08]. O protocolo está composto por duas camadas: a camada de gravação (*TLS Record Protocol*) e a camada de *Handshake (TLS Handshake Protocol)* que podem ser vistas na Figura 4.5.

Existem poucas diferenças entre os protocolos SSL v3 e TLS v1. A principal delas é que o TLS utiliza o algoritmo *Keyed-Hashing for Message Authentication Code (HMAC)*, enquanto que o SSL v3 utiliza o algoritmo *Message Authentication Code (MAC)*. O HMAC gera um valor para a verificação da integridade assim como o MAC gera, mas com a construção utilizando a função de *hash* torna-se muito mais difícil de ser quebrada e violada.

4.1.5 L2TP

Projetado pela *Cisco Systems* e, posteriormente, homologado pelo IETF como protocolo padrão, o L2TP baseia-se no *Layer Two Forwarding* (L2F) para solucionar os problemas do PPTP, sendo considerado o seu herdeiro [33]. Algumas características, como a utilização do PPP para fornecer o acesso remoto e a operação em ambientes como o NetBEUI e o IPX, são mantidas do PPTP. Da mesma forma que o PPTP, o L2TP é um protocolo baseado no PPP.

Uma das diferenças entre o L2TP e o PPTP está no protocolo utilizado na camada inferior. Enquanto o PPTP deve ser sempre utilizado acima do IP, o L2TP pode utilizar um conjunto de outros protocolos inferiores, como o PPP, o IP e o *Frame Relay*.

Sob o ponto de vista da segurança da comunicação, o L2TP, diferentemente do PPTP, não possui serviços de cifragem e integridade de dados, porém as informações iniciais, relativas ao processo de autenticação dos dois extremos do túnel são protegidas, enquanto que no PPTP os parâmetros podem ser livremente obtidos.

Outra diferença visível em relação a seu predecessor é quanto a forma de autenticação, pois ela é feita em dois níveis, no primeiro, o usuário é autenticado pelo provedor de acesso antes do túnel ser instalado e, no segundo, quando a conexão é estabelecida entre os roteadores [23].

Sendo um protocolo padrão, qualquer fabricante pode criar produtos que utilizem o L2TP, de forma que provedores de acesso e consumidores em geral não dependam de produtos fornecidos por uma única empresa.

Apesar de ser atual, o L2TP apresenta como desvantagem não possuir um mecanismo eficiente de encapsulamento, ou seja, para executar esta tarefa, ele necessita do protocolo *IPSec*, que será explicado posteriormente, para que juntos possam garantir a segurança da VPN [23] [34].

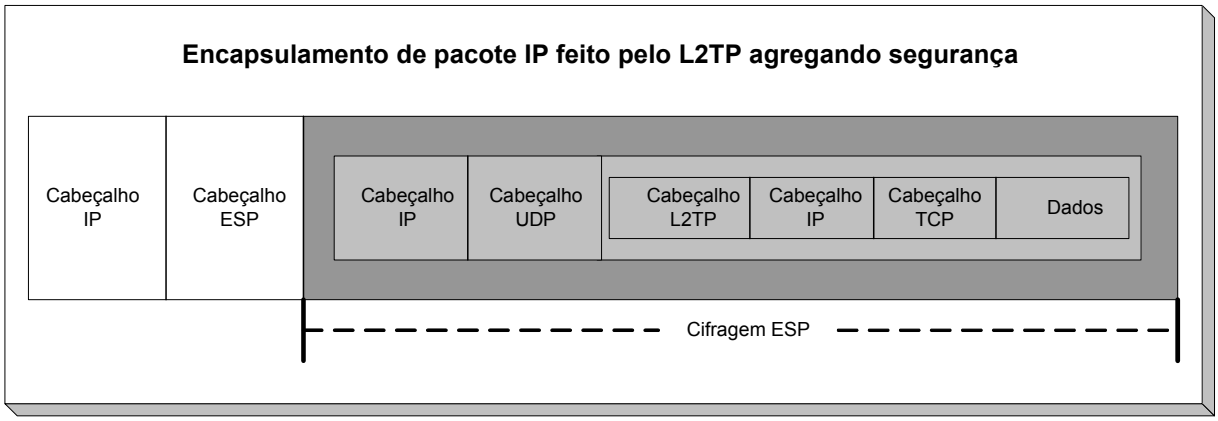

Figura 4.6 – Encapsulamento de um pacote IP feito pelo L2TP sob a proteção do cabeçalho ESP do *IPSec*.

Dado que o L2TP não foi projetado para a configuração de ambientes seguros, seu uso em cenários onde existe uma rede não confiável, como a Internet, entre os extremos de um túnel, deve sempre ser combinado com outros protocolos capazes de suprir a sua ausência de serviços de segurança. Um conjunto de propostas tem sido desenvolvido para conciliar o uso do L2TP com o *IPSec*. Quando executado sobre o IP, o L2TP é transportado através do UDP, desta forma, a aplicação da proteção do *IPSec* sobre o L2TP pode basear-se simplesmente no uso de seletores que filtram o tráfego L2TP.

É importante notar que a combinação do tunelamento do *IPSec* com o L2TP resulta na criação de dois túneis, um para cada protocolo. A Figura 4.6 baseada em [23] exibe o encapsulamento de um pacote TCP feito pelo L2TP sendo utilizado sobre o IP protegido pelo ESP[1].

Um problema na integração do L2TP com o *IPSec* é a impossibilidade do segundo levar em consideração os valores dos campos de pacotes IP encapsulados pelo primeiro [33]. Outros

[1] ESP: *Encapsulated Security Payload*

procedimentos de interação entre os dois protocolos têm sido sugeridos no intuito de prover o desenvolvimento de soluções para aspectos ainda não padronizados do *IPSec*. Uma das propostas sugere, por exemplo, o L2TP como protocolo para a troca de informações relativas às políticas de segurança entre um *host* e um roteador *IPSec*.

Apesar destas soluções serem práticas e de baixo custo, pelo fato do protocolo L2TP já ser um padrão definido, existem críticas severas quanto ao uso de um protocolo que não foi projetado para ambientes seguros na execução de procedimentos vitais para um protocolo de segurança como o *IPSec*.

4.1.6 *IPSecurity* (*IPSec*)

O projeto *IPSec* representa um esforço desenvolvido pelo *Working Group IPSec* do IETF para desenvolver uma arquitetura de segurança para o protocolo IP e tem como objetivos [07] [05]:

- Criar uma infra-estrutura de rede segura providenciando proteção nos cabeçalhos de dados e de chaves;
- Reduzir a preocupação de implementar mecanismos de segurança nas aplicações;
- Compatibilizar o seu funcionamento com mecanismos de segurança já existentes e utilizados por aplicações;
- Evitar problemas de exportação de criptografia;
- Ser parte integrante do protocolo IPv6 e poder ser aplicável ao IPv4.

Através dos seus componentes, o *IPSec* usa este conceito para permitir a implementação de redes virtuais privadas (VPN´s) e seguras através de redes públicas tais como a Internet [07] [05], representa uma arquitetura para o protocolo IP, integrando mecanismos de autenticação, gestão e distribuição de chaves que podem ser usados com qualquer das versões do protocolo IP.

São utilizados como mecanismos de autenticação dois cabeçalhos de extensão específicos do protocolo IPv6: o cabeçalho de Autenticação (*Authentication Header*) e o cabeçalho de encapsulamento de dados de segurança (*Encapsulating Security Payload Header*). Além destes dois cabeçalhos, o *IPSec* define também o conceito de **associação de segurança** –que é um conjunto de diretivas que permite negociar algoritmos de criptografia que irá utilizar.

Uma associação de segurança representa uma relação entre duas ou mais entidades que se comunicam e descrevem quais os mecanismos de segurança deverá utilizar para estabelecer uma comunicação segura. A associação de segurança permite negociar protocolos, algoritmos de criptografia e chaves a usar e contém informação sobre:

- Algoritmo e modo de autenticação que é aplicado ao cabeçalho de autenticação;
- Chaves usadas no algoritmo de autenticação;
- Algoritmo, modo e criptografia utilizados no cabeçalho de encapsulamento de dados de segurança, ESP;
- Chaves usadas no algoritmo de criptografia do cabeçalho de encapsulamento de dados;
- Chaves de autenticação usadas com o algoritmo que faz parte da transformada ESP;
- Tempo de vida da chave;
- Tempo de vida da associação de segurança;
- Endereço(s) fonte da associação de segurança;
- Nível de sensibilidade dos dados protegidos.

Na prática, uma associação de segurança é representada por um índice de parâmetros de segurança - *Security Parameter Index* (SPI) - com um endereço IP destino. O SPI é um campo que surge nos cabeçalhos de segurança IPv6 (AH e ESP), que não é criptografado na transmissão, já que a sua informação é essencial para decifrar a informação transmitida.

Quando uma entidade quiser estabelecer uma associação de segurança, utiliza um SPI e um endereço destino (pertencente a entidade com que deseja estabelecer comunicação segura) e envia essa informação à entidade com que quer estabelecer o canal seguro, assim, para cada sessão de comunicação autenticada entre dois nós, são necessários dois SPI - um para cada sentido, dado que cada associação de segurança é unidirecional.

O *IPSec* apresenta uma estrutura bastante flexível, que não obriga a utilização de algoritmos de autenticação ou criptografia específicos, deste modo o *IPSec* pode interagir com as normas mais recentes. No entanto, dada a necessidade de segurança, o IETF definiu alguns algoritmos para serem utilizados:

- HMAC-MD5 e HMAC-SHA-1 para autenticação (quer no cabeçalho AH, quer no ESP);
- DES-CBC[1], para a criptografia usada no cabeçalho ESP.

O *IPSec* integra gestão manual de chaves. A gestão é da responsabilidade de protocolos criados para este fim, tais como o SKIP, da *Sun Microsystems*, ou o *Photuris*, (acrônimo em latim para o desenvolvido por *Phil Karn*, ou ainda o protocolo *Internet Key Exchange, IKE*) [07].

Na medida em que estes cabeçalhos são cabeçalhos de extensão que irão ser adicionados a um cabeçalho IP, os encaminhadores podem interpretá-los como fazendo parte integrante dos dados, o que

[1] DES-CBC: *Data Encryption Standard-Cipher Block Chaining*

permite a compatibilidade destes mecanismos com equipamento que compreende o protocolo IP mas não o *IPSec*.

Os componentes da *IPSec* são:

- Cabeçalho de Autenticação (AH)
- Cabeçalho de Encapsulamento de Dados de Segurança (ESP)
- Mecanismos de Gestão de Chaves

4.1.6.1 Cabeçalho de Autenticação (AH)

O cabeçalho de autenticação, apresentado na Figura 4.7, representa um cabeçalho de extensão do protocolo IPv6 e foi criado para validar a identidade de entidades que estão se comunicando, ou seja, identifica o emissor e destino corretos. Deste modo pode ser utilizado para verificar se o emissor que afirma ter enviado os dados é exatamente quem afirma ser [07].

Este cabeçalho foi desenvolvido de modo a providenciar mecanismos de autenticação aos datagramas IP. Porém este cabeçalho por si só não fornece proteção contra ataques de análise de tráfego ou confidencialidade, sendo para tal usado normalmente em conjunto com o cabeçalho de encapsulamento de dados.

Próximo Cabeçalho	**Tamanho do Módulo**	**Reservado**
Índice de Parâmetros de Segurança (SPI)		
Número de seqüência		
Dados de Autenticação		

Figura 4.7 – Cabeçalho de autenticação (AH)

4.1.6.2 Cabeçalho de Encapsulamento de Dados de Segurança (ESP)

O cabeçalho de encapsulamento de dados de segurança (ESP), mostrado na Figura 4.8, é um cabeçalho de extensão pertencente ao protocolo IPv6 que fornece integridade e confidencialidade aos datagramas IP através da cifra dos dados contidos no datagrama. É responsável pela criptografia dos dados e é inserido entre o cabeçalho IP e o restante do datagrama. Desta forma, os campos de dados são alterados após serem criptografados. Juntamente com o ESP, segue o SPI para informar ao recipiente do pacote como proceder para abertura apropriada do conteúdo do mesmo.

Um contador no ESP informa quantas vezes o mesmo SPI foi utilizado para o mesmo endereço IP de destino. Esse mecanismo previne um tipo de ataque no qual os pacotes são copiados e enviados fora de ordem, confundindo assim os nós de comunicação. Todo o restante do pacote, com exceção a parte de autenticação, é criptografado antes de ser transmitido. Os algoritmos de criptografia mais utilizados são o DES (*Data Encryption Standard*) e o 3DES (*Triple Data Encryption Standard*) e protocolos proprietários de fabricantes.

O ESP também pode ser utilizado para autenticação, com o campo opcional destinado para esse fim. O somatório de verificação (*checksum*) é computado sobre todo o ESP, com exceção do campo de autenticação e o seu comprimento varia de acordo com o algoritmo utilizado. A autenticação do ESP é diferente da fornecida pelo AH, porque não protege o cabeçalho IP que precede o ESP, embora proteja um cabeçalho IP encapsulado no modo Túnel. O AH, por sua vez, protege este cabeçalho externo, juntamente com todo o conteúdo do pacote ESP. As duas autenticações não são utilizadas simultaneamente por questão de economia de processamento. A utilização do ESP pode ser efetuada de dois modos:

- **Modo de Transporte (*transport-mode*)**. Provê proteção principalmente no que tange aos protocolos da camada superior. É utilizado na maioria dos casos em comunicações ponto-a-ponto entre dois nós, por exemplo, um cliente e um servidor.

Este modo criptografa a informação do protocolo da camada de transporte, adicionando-lhe em seguida um novo cabeçalho IP não-criptografado, deste modo torna-se vantajoso em redes relativamente pequenas, nas quais o(s) servidor(es) e nó implementam o *IPSec* ;

- **Modo de Túnel (*tunnel-mode*)**. Provê proteção ao pacote IP. Para tal, após a adição dos campos ESP ao pacote IP, todo o pacote é tratado como o módulo de dados de um novo pacote IP. Assim, pode ser usado para enviar dados criptografados através de um túnel, o que permite enviar dados independentemente da infra-estrutura utilizada.

Um exemplo é o envio de pacotes IP através de canais virtuais criados numa rede IP pública, como a Internet. Através deste modo, pode ser fornecida segurança a um grupo de nós que não implementem o *IPSec*.

Índice do parâmetro de segurança (SPI)		
Vetor de inicialização		
Dados de *payload*		
padding	**Medida do *pad***	***Payload type***

Figura 4.8 – Formato do cabeçalho de encapsulamento de dados de segurança (ESP)

A Figura 4.9 demonstra os componentes de um pacote original IP e os modos de transporte e túnel.

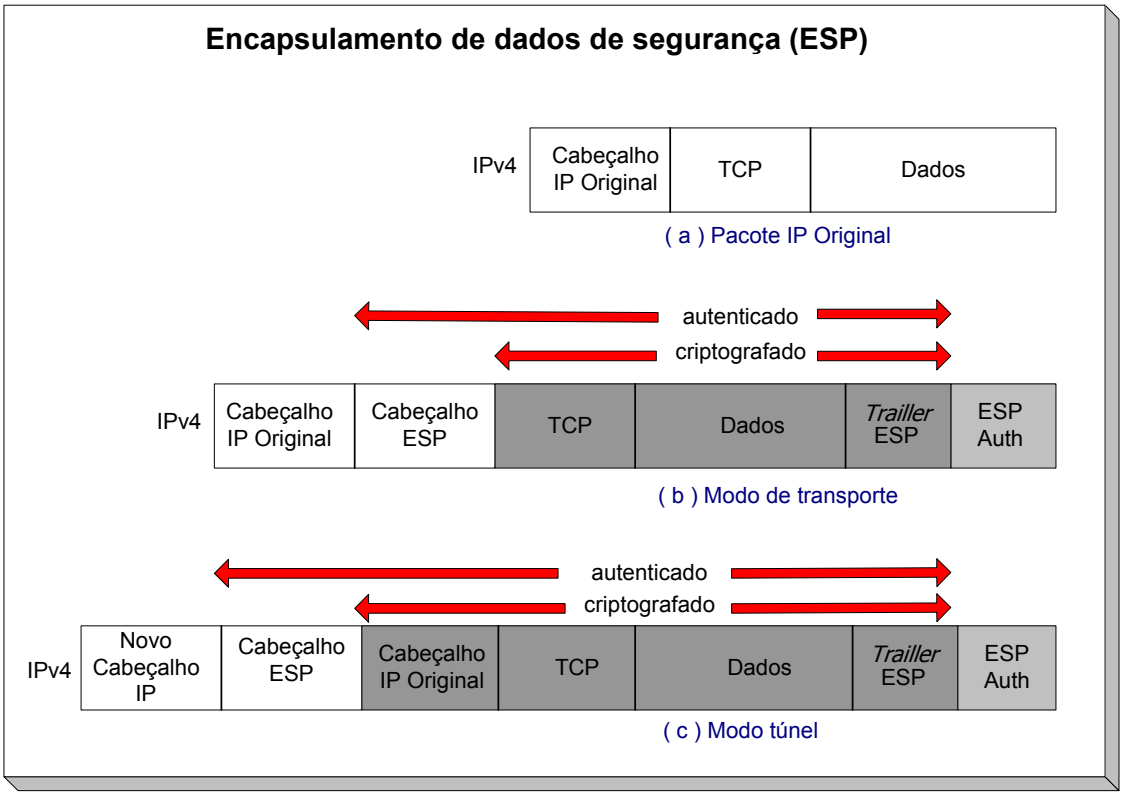

Figura 4.9 – Componentes dos pacotes em modo IP original, transporte e túnel em ESP

4.1.6.3 Mecanismos de Gestão de Chaves

Além dos mecanismos de autenticação e validação da informação o *IPSec* necessita de um mecanismo eficiente de gestão de chaves. A gestão de chaves diz respeito a criação, eliminação e alteração das chaves. Embora o *IPSec* não integre um mecanismo de gestão de chaves, o IETF definiu como norma de gestão o protocolo híbrido *ISAKMP/Oakley* também denominado IKE, *Internet Key Exchange*, que se encontra baseado nos documentos [07]:

- **ISAKMP** - *Internet Security Association and Key Management Protocol*. Protocolo que descreve uma infra-estrutura para a gestão de associações de segurança;
- ***Oakley*** - protocolo que define o conjunto de chaves para cifra, *hashing* e autenticação e é compatível com a gestão de associações de segurança ISAKMP;
- ***Internet Domain of Interpretation*** - define parâmetros ISAKMP para as associações de segurança *IPSec* no domínio Internet;

- **Resolução ISAKMP/Oakley** - define o perfil do protocolo híbrido ISAKMP/*Oakley*, escolhido como norma de gestão de chaves criptográficas pela *Internet Engineering Task Force*;

- **IKE** - *Internet Key Exchange*. O IKE utiliza a porta 500 UDP para interagir com os demais mecanismos de segurança *IPSec* através de associações de segurança para diversos protocolos e associações de segurança.

Desta forma permite uma utilização transparente para associar diferentes mecanismos de segurança sem envolver as entidades participantes na comunicação. O IKE agrupa funcionalidades dos protocolos ISAKMP (mensagens) e *Oakley* (modos). Quando uma entidade pretende estabelecer comunicação segura, passa pelas fases IKE que são:

- **Fase 1**: esta fase ocorre num meio inseguro. Tem o objetivo de estabelecer um canal seguro que irá proteger as trocas da Fase 2. É executada uma vez para várias fases 2;

- **Fase 2**: esta fase ocorre no canal seguro criado na fase 1. As suas negociações têm o objetivo de estabelecer as associações de segurança que irão proteger a comunicação.

Após estas duas fases, encontra-se estabelecido um canal seguro através do qual se pode efetuar comunicação segura. Existem ainda outros protocolos de gestão de chaves com os quais o *IPSec* pode interagir, por exemplo o SKIP e o *Photuris* [07].

4.2 Protocolos de autenticação

A autenticação é importante quando uma corporação oferece acesso em sua rede privada, através de uma rede pública como a Internet a funcionários que estão em trânsito, e que, precisam acessar a rede para atualizar ou consultar informações vitais [32].

O usuário distante, com o cliente de autenticação instalado em seu computador, tenta uma conexão com um endereço dentro da rede protegida pelo mecanismo servidor de autenticação, esse por sua vez, verifica que o computador remoto tem o cliente de autenticação e que possuí uma regra válida na estratégia de segurança, subseqüentemente, o servidor fornece o acesso, entretanto, o acesso é válido para um período limitado de tempo, depois do qual o processo de autenticação será requerido novamente.

Neste trabalho comenta-se sobre dois deles, o RADIUS e o KERBEROS, pois tratam-se dos principais métodos utilizados atualmente em produtos comerciais, principalmente quando estamos trabalhando com plataformas UNIX ou *Windows*.

4.2.1 Protocolo RADIUS

Baseado em um modelo de segurança distribuída previamente definido pelo IETF, o RADIUS provê um sistema de segurança Cliente/Servidor aberto e escalável. O servidor RADIUS pode ser adaptado facilmente para trabalhar com produtos de segurança de terceiros ou em sistemas de segurança proprietários. Qualquer mecanismo de comunicação, seja um software ou um hardware que utilize o protocolo cliente RADIUS pode se comunicar com um servidor RADIUS [32].

O RADIUS autentica através de uma série de comunicações entre o cliente e o servidor. Uma vez que o usuário é autenticado, o cliente proporciona a ele, o acesso aos serviços apropriados. Os passos envolvidos no processo do RADIUS podem ser descritos da seguinte forma:

O *PortMaster*[1] cria um pacote de dados com as informações e o chama de "pedido de autenticação". Este pacote inclui a informação que identifica o *PortMaster* específico que envia o pedido de autenticação, a porta que está sendo usada para a conexão de modem, identificação do usuário e a senha. Para proteger os dados de *hackers* que possam estar escutando a conexão, o *PortMaster* age como um cliente RADIUS e codifica a senha antes que seja enviada em sua jornada ao servidor RADIUS .

Quando um pedido de autenticação é recebido, o servidor de autenticação valida o pedido e então decifra o pacote de dados para ter acesso a identificação do usuário e senha. Esta informação é passada para o sistema de segurança apropriado. Se o usuário e senha estiverem corretos, o servidor envia um reconhecimento de autenticação que inclui informação sobre o usuário e as exigências dos serviços.

Por exemplo, o servidor RADIUS contará para o *PortMaster* que um usuário precisa do Protocolo PPP (ponto-a-ponto) para se conectar a rede. O reconhecimento pode também, conter filtros, com informações sobre os limites de acesso do usuário para os recursos específicos na rede. Se o usuário e a senha não estiverem corretos, o servidor RADIUS envia um sinal ao *PortMaster* e o usuário terá o acesso negado a rede.

Uma vez que a informação é recebida pelo *PortMaster*, o servidor RADIUS envia uma chave de autenticação, ou assinatura, se identificando para o cliente RADIUS e permitindo então, a configuração necessária para que os serviços de envios e recepções personalizados, funcionem para o usuário autenticado.

[1] *Portmaster*: É usualmente utilizado para autenticação RADIUS, pode ser um hardware específico (como *Livingston RADIUS Server*) ou software através de uma máquina rodando UNIX.

4.2.2 Protocolo KERBEROS

KERBEROS é um serviço de autenticação distribuído que permite que um cliente, através de um usuário, forneça sua identidade a um servidor de autenticação, passando em seguida por um verificador de sessão, para que então, estabeleça a transferência das informações com o *host* destino, evitando assim, a violação da conexão estabelecida. Esse protocolo foi desenvolvido em meados dos anos 80 como parte do Projeto do *MIT Athena*. Hoje em dia, é uma das soluções para os problemas de segurança em rede, pois fornece ferramentas de autenticação e criptografia para trabalhos em redes públicas como a Internet [32].

Muitos dos protocolos usados na Internet não provêem segurança. Ferramentas que varrem senhas fora da rede são usadas em brechas de sistemas. Assim, aplicações que enviam senha sem criptografia pela rede Internet são extremamente vulneráveis. Contudo, em muitas aplicações Cliente/Servidor que são desenvolvidas e implementadas, não é dada a devida atenção sobre os aspectos de segurança e autenticação. Alguns administradores tentam usar *Firewalls* para resolver os problemas de segurança de rede. Infelizmente, os *Firewalls* assumem que os acessos ruins estão todos do lado de fora da rede, o que freqüentemente é uma suposição muito ruim.

Existem muitos usuários e colaboradores em trânsito, que em geral, possuem restrições para usar a rede interna, pois os mecanismos de segurança vão descartar suas tentativas de acesso não autorizadas.

O sistema KERBEROS usa ingressos eletrônicos para autenticar um usuário para um servidor. Um ingresso só é bom para um único servidor e um único usuário durante um certo período de tempo e para uma mensagem codificada que contém o nome do usuário, o seu servidor, o endereço da rede do servidor do usuário, um selo de tempo e uma chave de sessão, uma vez que o usuário adquire este ingresso, ele pode usar isto para ter acesso ao servidor quantas vezes forem necessárias até que o ingresso se expire, o usuário não pode decifrar o ingresso, mas pode apresentá-lo ao servidor. Com isso, escutas clandestinas não podem violar o ingresso quando este estiver em curso na rede Internet [32].

O protocolo KERBEROS envolve dois servidores, um de autenticação e o outro TGS[1] que concede os ingressos. Os passos envolvidos no processo do protocolo KERBEROS estão descritos a seguir.

[1] TGS: *Ticket-Granting Service*

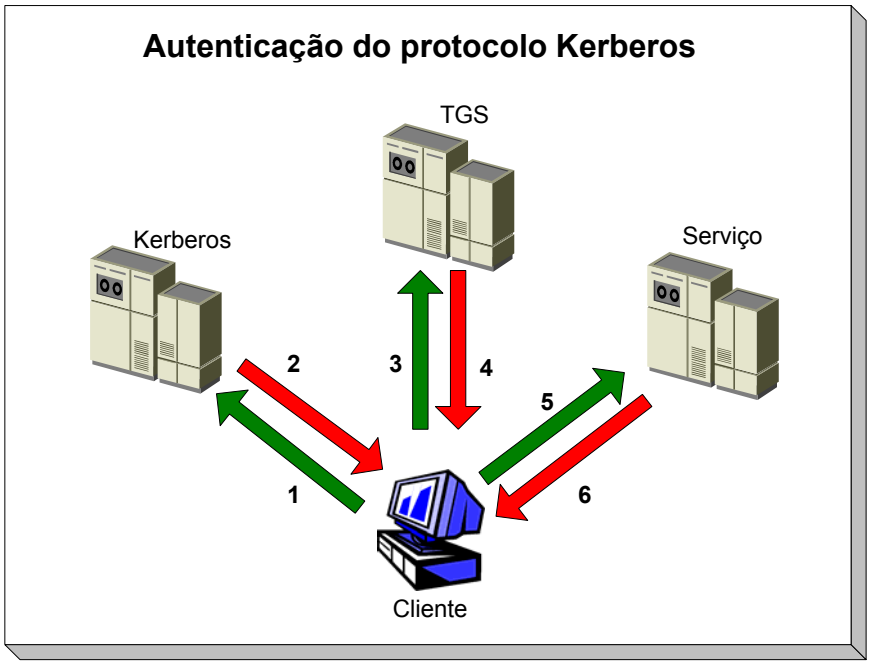

Figura 4.10 – Passos no processo da autenticação do protocolo Kerberos

1) Obter um ingresso para um servidor designado. O usuário primeiro pede ao servidor de autenticação KERBEROS um ingresso para o KERBEROS TGS. Este pedido leva a forma de uma mensagem que contém o nome do usuário e o nome do TGS (pode haver vários);

2) O servidor de autenticação verifica o usuário em seu banco de dados e então gera uma chave de sessão para ser usada entre o usuário e o TGS. KERBEROS codifica esta chave de sessão que usa a chave de segredo do usuário (processo de uma só direção com senha do usuário). Então cria um TGT[1] para o usuário apresentar ao TGS e codifica o TGT usando a chave de segredo do TGS (que só é conhecido pelo servidor de autenticação e o servidor TGS). O Servidor de Autenticação envia de volta as mensagens codificadas ao usuário;

3) O usuário decifra a primeira mensagem e recupera a chave de sessão. Logo, o usuário cria um autenticador que consiste em seu nome, seu endereço de rede e um selo de tempo, tudo codificado com a chave de sessão gerada pelo servidor de autenticação KERBEROS. O usuário então envia o pedido ao TGS para fazer ingresso a um servidor designado. Este pedido contém o nome do servidor, o TGT KERBEROS (que foi codificado com o a chave de segredo do TGS) e o autenticador codificado;

[1] TGT: *Ticket-Granting Ticket*

4) O TGS decifra o TGT com sua chave secreta e então usa a chave de sessão incluída no TGT para decifrar o autenticador. Compara a informação do autenticador com a informação do ingresso, o endereço da rede do usuário com o endereço foi enviado no pedido e o tempo estampado com o tempo atual. Se tudo se emparelhar, permite a continuação do pedido. O TGS cria uma chave de sessão nova para o usuário e o servidor final com esta chave em um ingresso válido para o usuário apresentar ao servidor. Este ingresso também contém o nome do usuário, endereço da rede, um selo de tempo e um tempo de vencimento para o ingresso, que foi codificado com a chave de segredo do servidor designado e o seu nome. O TGS também codifica a nova chave de sessão designada que vai ser compartilhada entre o usuário e o TGS. Envia ambas as mensagens de volta ao usuário;

5) O usuário decifra a mensagem e a chave de sessão para uso com o servidor designado. O usuário está agora pronto para se autenticar com o servidor. Ele cria um autenticador novo codificado com a chave de sessão de usuário e servidor final que o TGS gerou.

Para pedir acesso ao servidor final, o usuário envia junto ao ingresso recebido de KERBEROS (que já é codificado com a chave de segredo do servidor designado) o autenticador codificado. O autenticador contém o texto plano codificado com a chave de sessão, prova que o remetente sabe a chave. Da mesma maneira é importante codificar o tempo, para prevenir que terceiros que tentem registrar o ingresso e o autenticador possam usar as informações em futuras conexões;

6) O servidor designado decifra e confere o ingresso e o autenticador, também confere o endereço do usuário e o selo de tempo. Se tudo confirmar, o servidor sabe agora que o usuário é que esta reivindicando o acesso é realmente ele. A partir deste instante podem usar a chave de criptografia para comunicação segura. (Como só o usuário e o servidor compartilham esta chave, eles podem assumir que uma recente mensagem codificou aquela chave originada com a outra chave anterior);

Para aplicações que requerem autenticação mútua, o servidor envia para o usuário uma mensagem que consiste no selo de tempo mais 1 (t+1), codificada com a chave de sessão. Isto serve como prova ao usuário que o servidor soube da sua chave secreta de fato e pôde decifrar o ingresso e o autenticador [32][1].

[1] Nota: O protocolo RADIUS é adequado em sistemas de serviços remotos discados, enquanto o KERBEROS, pode ser utilizado através de qualquer tipo de conexão. A autenticação é um dos pontos forte na segurança de qualquer sistema, pois tem a finalidade de atravessar os mecanismos de segurança para autenticar o usuário, autorizando ou não a sua conexão [32].

Conclusão

Os protocolos seguros apresentados neste Capítulo representam o resultado de um grande esforço realizado pelas diversas instituições de pesquisas, empresas e em muitos casos pessoais, com objetivo de proporcionar a realização de uma comunicação segura através de um canal vulnerável. Neste sentido pode-se afirmar que já existem ferramentas suficientes para ficarmos mais tranqüilos quando acessamos a rede local do trabalho utilizando um computador remoto para navegarmos na Internet, utilizar o correio eletrônico ou mesmo aplicações que acessam a base de dados.

A segurança da informação dependerá de um conjunto de iniciativas que tanto o administrador quanto o cliente dos sistemas tenham condições e possam aplicar e conhecer estes mecanismos.

Capítulo 5

Soluções de segurança para aplicações Cliente/Servidor

No capítulo anterior vimos que os protocolos se modernizaram para agregar segurança ao meio. Deste modo pôde-se criar mecanismos de baixo custo para integrar redes geograficamente distantes, atendendo aos anseios de boa parte das empresas e instituições. Entretanto apenas os protocolos seguros não fornecem todos os subsídios para que as aplicações do tipo Cliente/Servidor tenham sucesso. É necessário atender também as limitações impostas pelos Sistemas Operacionais, sejam eles livres ou proprietários, utilizados nas estações de trabalho ou em servidores da rede.

Para exemplificar questões relacionadas a segurança da informação e autenticidade do usuário, será apresentado neste Capítulo um estudo de caso real, utilizando uma aplicação comercial do tipo *ERP* gerado para administrar informações relacionadas a Área Administrativa que manipula dados principalmente de Recursos Humanos.

Este software foi desenvolvido por uma empresa brasileira com mais de 500 clientes espalhados pelo País. O produto está baseado na arquitetura Cliente/Servidor e faz acesso ao banco de dados relacional.

Em seguida mostraremos algumas das técnicas utilizadas para interligar a parte cliente de uma aplicação aos seus respectivos servidores.

Finalmente são expostas quais as soluções encontradas para que estas aplicações possam executar suas tarefas, independente da tecnologia de enlace físico utilizado, tendo garantido os requisitos de compatibilidade entre os Sistemas Operacionais dos envolvidos na comunicação, a segurança da informação e também da identidade através da autenticação do usuário.

5.1 Um estudo de caso

Nos Capítulos anteriores foram definidos e apresentados os elementos que são envolvidos numa comunicação do tipo Cliente/Servidor, partiremos para uma análise prática tendo como base *softwares* desenvolvidos por empresas para fins comerciais e utilizados principalmente nas áreas administrativas de empresas privadas ou públicas.

O Produto utilizado em nosso estudo de caso é um pacote de aplicativos, desenvolvido no Brasil e de propriedade de uma empresa nacional. É composto por arquivos do tipo binários executáveis, arquivos de ajuda, arquivos de parametrização entre outros e, cada módulo possui funcionalidades específicas.

Para facilitar o entendimento da composição do Produto, pode-se comparar sua estrutura com o Produto *Office* da *Microsoft*, pois ele é bastante conhecido. O *Microsoft Office* é um conjunto de aplicações modulares e com funções definidas como Editor de Texto, Planilhas de Cálculo Eletrônico, *Software* para Apresentações e outros menos conhecidos que se inter-relacionam.

O Produto de nosso estudo de caso, estruturalmente possui o mesmo conceito do *MS-Office*. As funcionalidades estão divididas em módulos, mas seus aplicativos possuem tarefas muito mais complexas. São aplicações que manipulam informações de caráter gerencial e na maioria dos casos sigilosas, como por exemplo: Administrar os dados do funcionário da empresa/instituição, calcular a folha de pagamento com base na legislação do país e processos relacionados a Administração de Pessoal. Também possui funcionalidades como gerar arquivos com dados da empresa e funcionários para envio à órgãos federal e estaduais, fazer o controle de treinamentos da empresa, administrar os recursos financeiros, emitir relatórios gerenciais entre outras tarefas. Portanto trata-se de um produto de administração de Recursos Humanos.

É importante citar o fato que a empresa mantém duas versões de seus aplicativos. Além da versão Cliente/Servidor, existe a versão para WEB, porém esta não possui todas as funcionalidades e facilidades que a primeira oferece.

A versão WEB é utilizada principalmente para consultas aos dados pessoais e disponibiliza poucas rotinas destinadas a manutenção administrativa dos dados, como entrada de dados de freqüência de funcionários, relatórios administrativos simples entre outros.

Por uma questão profissional não serão divulgados os nomes das empresas fornecedoras dos softwares, pois o objetivo deste trabalho é acadêmico, dando ênfase aos conceitos que envolvem a segurança e apresentação de soluções para este problema na plataforma de execução da aplicação.

A Tabela 5.1 apresenta as principais características do produto.

Tipo de Arquitetura Cliente/Servidor	Centrada no Cliente (Figura 2.3) para versão Cliente/Servidor
Linguagem de programação do produto	*Borland Delphi 5.0*
Gerenciadores de Banco de Dados Suportados	*Oracle, IBM DB2* ou *Informix*
Comunicação com Banco de Dados	Utiliza-se da BDE[1] instalada no cliente
Parametrização do produto	Realizada através de arquivos armazenados no servidor
Protocolo de comunicação em Rede	TCP/IP e NETBEUI (sobre TCP/IP)
Plataforma utilizada para Clientes	*Windows*
Plataforma utilizada para Servidores	*Windows 2000/2003 Server* (arquivos do produto) e/ou UNIX com Apache *WebServer* para arquivos WEB
Observação: Existe também a versão WEB para este produto, porém esta não possui todas as funcionalidades da versão C/S e tão pouco funciona em qualquer plataforma, mesmo que trabalhe com o protocolo TCP/IP (Internet). Isso se explica pelo fato das páginas conterem códigos utilizando tecnologia CSS[2] e também controles *ActiveX*[3].	

Tabela 5.1 – Descrição do Produto utilizado no estudo de caso

Na Figura 5.1 é demonstrado o cenário real mínimo de comunicação da versão Cliente/Servidor do produto.

[1] BDE: *Borland Database Engine*
[2] CSS: (*Cascading Style Sheets*), são modelos desenvolvidos para que possibilitam controlar e melhorar a apresentação e o layout de elementos HTML nas páginas WEB
[3] ActiveX: São componentes de software definidos para interagir entre com outros componentes com tecnologia Java ou *Microsoft Component Object Model* (COM) em um ambiente de rede

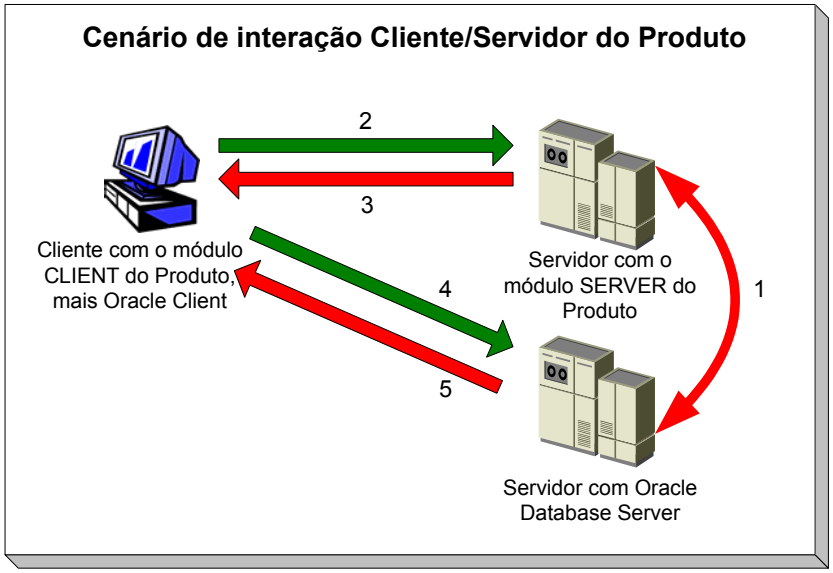

Figura 5.1 – Cenário mínimo do Produto versão Cliente/Servidor

A descrição completa da seqüência de atividades necessárias pela aplicação, apresentadas na Figura 5.1, estão expostas na Tabela 5.2.

Etapa	Atividade
1	Ao ser iniciado, o módulo *SERVER* inicia alguns serviços personalizados instalados no *Windows* 2000/2003 *Server*, dentre eles estão o serviço que verifica a conexão com o Banco de Dados e o gerenciador de conexões dos clientes, que fica aguardando requisições dos clientes do Sistema, pois somente se estes serviços estiverem ativos é que os clientes conseguirão se comunicar e ter acesso aos arquivos e a base de dados pelo módulo Cliente
2	Com o módulo *SERVER* já iniciado, o Cliente, através da versão CLIENT do produto instalado no *desktop*, envia os dados criptografados para autenticação no sistema (usuário/senha) através da rede
3	O módulo *SERVER* recebe as informações e verifica se o usuário/senha estão corretos com os dados armazenados na base de dados, bem como os privilégios que este usuário terá sobre o Produto. Nesta etapa podem ser desabilitadas algumas funcionalidades para restringir o uso
4	O banco de dados *Oracle Client* instalado no cliente abre uma sessão com o servidor de banco de dados através dos parâmetros definidos na BDE. Por este canal serão transmitidas todas as informações que são requisitadas através de comandos SQL do Produto ao servidor de banco de dados

Etapa	Atividade
5	O servidor do banco de dados atende às solicitações do módulo Client através do canal estabelecido

Tabela 5.2 – Descrição das Etapas do Cenário de interação do produto

Uma observação importante: É utilizado como exemplo neste estudo de caso o banco de dados *Oracle 9i*, lembrando que este pode ser substituído por qualquer um outro mostrado na Tabela 5.1.

A Tabela 5.3 mostra quais as funcionalidades exercidas em cada um dos módulos do produto.

Cliente	Servidor
Armazena uma cópia dos arquivos Binários Executáveis dos módulos e demais associados (*.ini, *.hlp etc). Quando há atualizações destes arquivos no Servidor, no momento da primeira execução eles são novamente copiados para os clientes. Todo o processamento (cálculo de folha de pagamento, fechamento de freqüência dos funcionários, execução de relatórios e outros) **ocorre NO CLIENTE**, portanto é um produto com arquitetura centrada no cliente (Figura 2.3). As informações processadas pelo Servidor de Banco de Dados são transferidas diretamente ao Cliente. O Cliente precisa acessar os arquivos do Produto através de uma conexão de rede que possua o protocolo NETBEUI (nativo ou sobre TCP/IP)	Armazena os arquivos fundamentais para execução do Sistema nos Clientes. Estes são divididos em diretórios com funcionalidades específicas, como por exemplo: • Modelos de Relatórios (da Empresa e Personalizados) • Regras para cálculos em geral (folha de pagamento, freqüência de funcionários, entre outros) • Modelos de Telas do Sistema (da Empresa ou Personalizados) • Arquivos Binários Executáveis do Sistema (*.exe) e demais arquivos associados aos módulos (*.ini, *.hlp e outros) É responsável por manter os serviços de conexão e abertura e fechamento das sessões com os Clientes do Produto. Deve permitir que os clientes tenham acesso a estes arquivos através da rede, esta tarefa é realizada utilizando-se o protocolo NETBEUI (nativo ou sobre TCP/IP)

Tabela 5.3 – Descrição das atribuições do Cliente e do Servidor no produto

Por se tratar de um produto voltado para instalação em plataformas *Windows 2000/2003 Server*, os arquivos do Sistema (parte Servidor) geralmente são instalados no mesmo servidor onde os Serviços do Produto estão. Opcionalmente pode-se ter uma configuração onde os usuários acessam os arquivos através de um outro Servidor UNIX que possua a funcionalidade de compartilhar arquivos para estações *Windows*, neste caso o software de domínio público SAMBA[1].

Na Figura 5.2 é demonstrado o cenário de interação adequado para implantação do produto com integração entre redes distintas. É importante observar que todas as interações da Figura 5.1, que também faz parte da solução, estão ocultas, mostradas apenas pelas setas verde e vermelha.

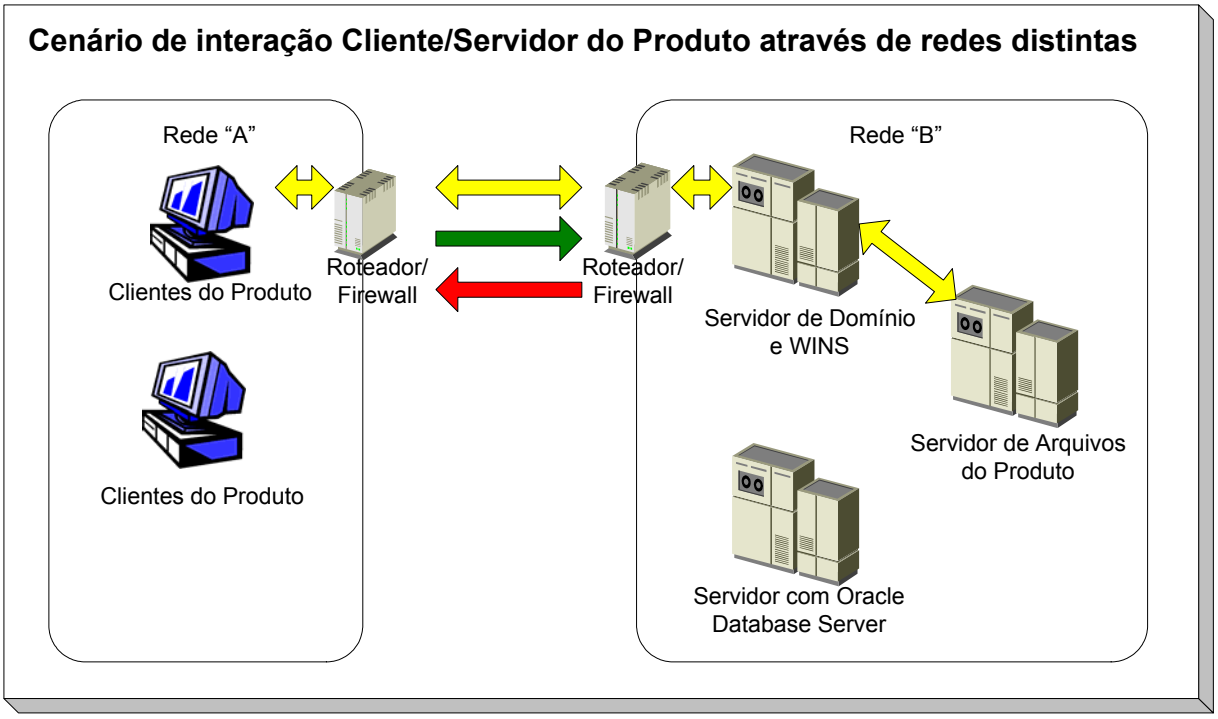

Figura 5.2 – Cenário do Produto versão Cliente/Servidor numa rede *Windows*

O problema estaria totalmente resolvido se não fosse a questão do tráfego de dados pela rede, dependendo da configuração do cenário (versão do banco de dados e sistema operacional no servidor, por exemplo) pode-se ter sérios problemas com os dados, que muitas vezes, estarão passando abertamente pela rede, sem nenhum tipo de criptografia. As soluções para este e outros problemas, serão discutidas mais adiante, no Capítulo 6, bem como as opções para solução completa do problema.

[1] SAMBA: Aplicação desenvolvida para possibilitar a utilização de servidores UNIX o uso de protocolo SMB, como conseqüência a integração com redes MS-Windows. http://www.samba.org

5.1.1 Análise de problemas encontrados no Cenário

Embora o cenário pareça simples, existem problemas relacionados a segurança da informação que necessitam serem analisados cuidadosamente, antes que o processo de implantação deste tipo de produto se inicie nas empresas ou em instituições públicas.

Obviamente não serão abordadas as questões dos valores referentes aos custos de manutenção de uma estrutura como a apresentada. O importante é expor o Produto do ponto de vista técnico, onde existem problemas no que tange ao uso do protocolo de comunicação padrão para acesso aos arquivos, sua utilização em uma rede de computadores e também a questão da segurança dos dados que irão trafegar durante esta comunicação.

Numa situação real, são raros os casos em que todos os elementos apresentados no cenário da Figura 5.1 façam parte da mesma rede local. Mesmo em tal situação, todos os usuários precisam estar autenticados sob um domínio *Windows* ou SAMBA, deste modo garante-se a identificação dos usuários do sistema através de seu SID[1] ou conta no servidor UNIX/SAMBA. Caso contrário, todos os clientes que precisarão acessar os arquivos compartilhados no servidor da aplicação, necessitarão ser autenticados novamente (pois já o são quando fazem o *logon* na estação de trabalho) através de uma nova conta, gerada pelo administrador do servidor. Esta opção torna-se inviável em todos os aspectos.

Do ponto de vista da segurança tem-se uma quantidade de contas no servidor indesejável, pois a medida em que novos usuários são cadastrados, maiores são as possibilidades para um atacante realizar suas investidas, aproveitando-se da vulnerabilidade do protocolo de comunicação principal.

A questão da administração torna-se mais complexa, pois além das tarefas usuais para manter a aplicação em funcionamento, terá como atribuição adicional a administração das contas de usuários e os serviços relacionados como criação de políticas de acesso, privilégios, rotinas para substituição de senhas perdidas entre outras, amplamente conhecidas pelos administradores de sistemas.

5.1.1.1 Vulnerabilidades do cenário – autenticidade do usuário

Para que a aplicação funcione de forma transparente, ou seja, sem que o usuário necessite se autenticar toda vez que precisa acessar os arquivos da aplicação que estão no servidor remoto, todos os usuários, que possuam a versão Cliente do Produto, devem ter condições de mapear uma unidade de

[1] SID: *Security Identifier*, vide item 6.1.1

disco remoto (como o NFS[1] no UNIX) no servidor de arquivos. Isso geralmente implica em uma reestruturação da rede, juntamente com o modo como os usuários são autenticados.

Por se tratar de uma aplicação executada apenas em estações de trabalho *MS-Windows*. É necessário que estes usuários sejam cadastrados e autenticados por um servidor de domínio *Windows NT 4.0* ou SAMBA. Para algumas empresas/instituições que estejam mais adiantadas e com conhecimentos definidos pode-se utilizar também a estrutura de Diretório Ativo presente na versão do *Windows 2000/2003 Server*.

Sem esta estrutura, um atacante pode simplesmente monitorar o canal de comunicação e capturar os pacotes que trafegam na rede com as informações para autenticação no servidor. Existe para isso uma série de ferramentas *shareware* e *freeware* disponíveis na Internet. Estes pacotes, caso não utilizem protocolos seguros, trazem informações valiosas como nome do usuário, nome do diretório compartilhado pelo servidor e em alguns casos a senha de acesso com a criptografia padrão das redes Microsoft. Um bom atacante possui ferramentas capazes de realizar ataques com a utilização de dicionários de senhas ou mesmo de força bruta, que é menos eficiente, pois faz uma análise combinatória de uma série de caracteres.

Não é intenção deste documento apresentar como estas ferramentas funcionam, mas em testes realizados, podem afirmar que são muito eficientes, principalmente quando utilizadas em redes mal estruturadas ou compartilhadas.

Se em uma destas contas ele conseguir o acesso, o que é provável se não houver uma política de senha que garanta parâmetros mínimos para tamanho de senha, por exemplo, ele pode ter acessos as informações da aplicação e ainda tornar-se um administrador do servidor, o que seria um desastre.

A solução para este problema é simples, com a utilização de técnicas que permitam a integração dos domínios *MS-Windows* e o uso de protocolos seguros. Estas técnicas e as soluções viáveis serão abordadas no Capítulo 6.

5.1.1.2 Vulnerabilidades do cenário – segurança na comunicação cliente/servidor

O segundo ponto a ser abordado na verdade é uma conseqüência do primeiro. Como observado anteriormente, dificilmente teremos todos estes elementos numa mesma rede local. Ocorre que a maioria das instalações prevê os servidores em redes distintas, protegida da melhor maneira possível, isso inclui mudanças nas regras dos *firewalls* nas entradas destas redes, pois agora eles precisam

[1] NFS: *Network File System*

permitir que as portas utilizadas pelo produto (incluindo NETBEUI e Banco de Dados), estejam liberadas para todos os clientes.

Mais uma vez o administrador se depara com problemas de segurança, pois as portas necessárias são conhecidamente fontes para ataques automatizados. *Sniffers,* como são conhecidos os programas que vasculham e "escutam" a rede, podem varrer todas as portas de um computador a procura de falhas ou brechas que permitam a instalação e execução de programas maliciosos.

Não se pode esquecer que a aplicação utiliza bancos de dados relacionais para efetuar suas transações, deste modo é necessário também configurar estes softwares para trabalharem com algoritmos de autenticação e criptografia, desta forma, independente do protocolo utilizado no meio de transmissão, os dados relativos a base de dados estão protegidos contra acessos indevidos e, informações utilizadas para conexão e execução de rotinas no banco de dados trafegam de forma ilegível. As medidas utilizadas para evitar o tráfego em claro são apresentadas no Capítulo 6 deste trabalho.

5.1.1.3 Vulnerabilidades do cenário – segurança nas estações de trabalho

O terceiro problema encontrado está no Cliente. Em nada adiantará possuir toda a segurança necessária no Servidor, se as estações não oferecem o mínimo de segurança. Por uma questão de caráter obrigatório, todas as estações precisam de Sistemas Operacionais da plataforma *MS-Windows* que por *default* trabalham com o protocolo NETBEUI. Neste sentido é importante lembrar que a maioria dos ataques que ocorrem nos Sistemas Operacionais *Windows 95/98* tem por objetivo capturar os arquivos ***.pwl**.

A razão é muito simples, quando um usuário conectado a um domínio ou não se identifica no *desktop* o Sistema Operacional gera automaticamente um arquivo **(usuário).pwl** e o grava no diretório padrão do Sistema Operacional, em geral C:\WINDOWS. O problema é que a criptografia utilizada neste procedimento é muito simples e existem dezenas de programas disponíveis na Internet, um exemplo é o PWLVIEW ou PWLTools (http://lastbit.com/vitas/pwl.asp), que podem resolver este problema para alguns *hackers* de plantão.

Em virtude destes fatos, não é apropriada a utilização destes Sistemas Operacionais. Aconselha-se utilizar as versões mais modernas do Sistema Operacional *Windows*, como o 2000 Profissional ou XP Profissional que implementam algoritmos criptográficos mais eficientes.

5.1.2 Análise da Aplicação versão WEB

No final da Tabela 5.1, foram apresentadas algumas características da versão WEB deste mesmo produto, o objetivo agora é mostrar com mais detalhes os elementos e o funcionamento desta versão.

A Figura 5.3 ilustra um cenário da versão WEB, colocando o *WebServer* e os arquivos do Produto em máquinas separadas, no entanto, é comum encontrar um ambiente em que as empresas, em geral, instalam toda a parte WEB do Produto em apenas um servidor, escolhendo como *WebServe*r o serviço *web* nativo da *Microsoft* denominado *IIS*[1]. Outro motivo que leva a esta configuração é o fato da customização dos serviços do produto, muito mais fácil que na versão com Unix/Apache, por exemplo.

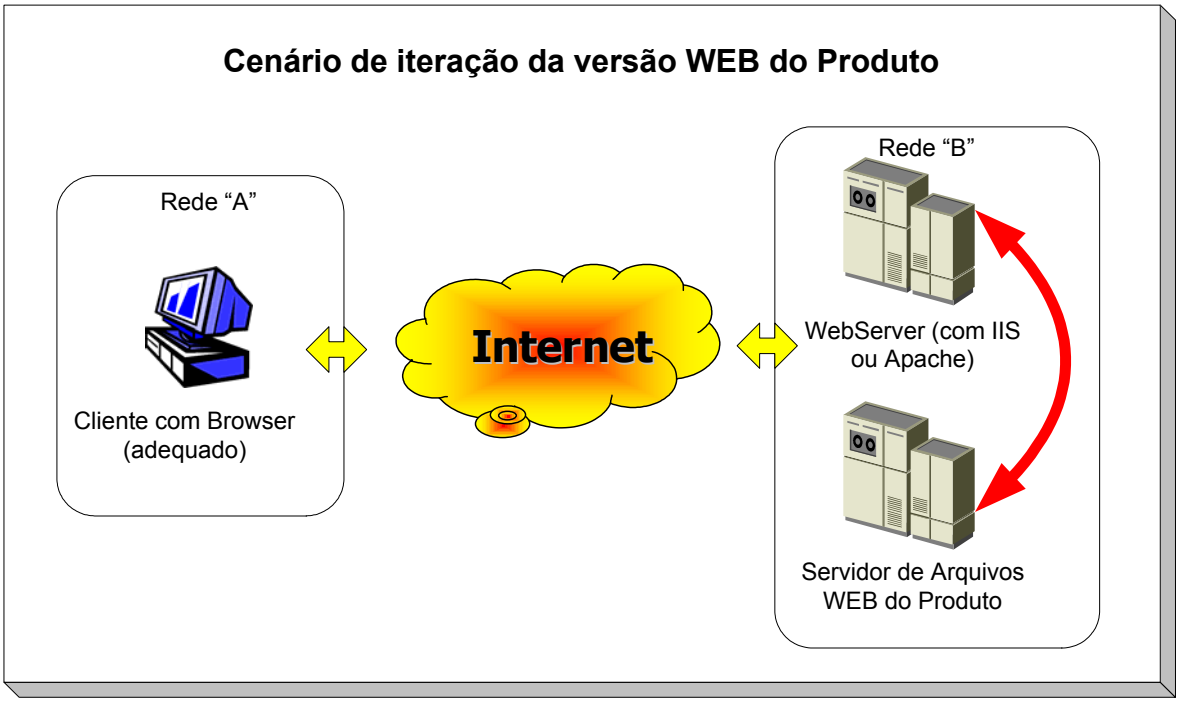

Figura 5.3 – Cenário do Produto versão WEB

A interação entre cliente e servidor do produto ocorre normalmente com o protocolo http, porém devido a algumas característica e tecnologias utilizadas na construção da versão WEB, tanto o cliente quanto o servidor devem satisfazer algumas premissas que são mostradas na Tabela 5.4.

[1] IIS: *Internet Information Service*

Cliente	Servidor
• Necessita de *browsers* que suportam as tecnologias de páginas CCS e controles ActiveX • Deve permitir o estabelecimento de conexões FTP de modo ativo para receber os relatório através do *browser*	• Devem possuir um software para publicação de página para Internet, geralmente o IIS da *Microsoft* ou o Apache (software Livre). • Necessita do serviço de FTP de modo ativo habilitado, pois esta é a maneira como os arquivos de relatório gerados pelo Produto são transmitidos aos *browsers* dos Clientes. • Armazenam os arquivos do Produto e também os temporários em um diretório que precisa ser compartilhado para uma rede *Windows*, pois o programa de atualização da versão do sistema precisa acessar estes arquivos para substituí-los. • Faz o controle da quantidade de instâncias ativas de cada módulo (informado pela empresa e conforme o contrato)

Tabela 5.4 – Descrição das atribuições do Cliente e do Servidor no produto para WEB

É evidente que o Produto, da maneira como está apresentado para WEB, terá problemas de segurança, mas neste momento o objetivo é apenas expor suas características. A solução para a versão WEB também será discutida no Capítulo 6 deste trabalho.

5.2 Tecnologias viáveis para solução do estudo de caso

Em vista do cenário apresentado, verifica-se a necessidade pela busca por soluções que atendam os seguintes requisitos:

- Segurança no tráfego entre o cliente e o servidor, tanto para a aplicação quanto para o acesso ao banco de dados.
- Autenticidade da identidade do usuário de forma a garantir o não repúdio
- Transparência para o uso da aplicação, escondendo-se a complexidade da estrutura montada.

Atualmente existe uma série de tecnologias que podem ser empregadas para a solução deste problema, as principais serão expostas a seguir e sua aplicação efetiva na solução tratada no Capítulo 6 desta dissertação.

5.2.1 VPN (Virtual Private Network)

VPN é a sigla das palavras *Virtual Private Network*. Pode-se definir VPN como uma rede de dados privada que utiliza a infra-estrutura pública de telecomunicações, mantendo a confidencialidade e integridade dos dados. Através de uma VPN cria-se uma conexão segura entre duas redes ou entre dois *hosts*, pelo encapsulamento dos pacotes dentro de um outro protocolo, formando um túnel onde os dados são transmitidos criptografados.

Em geral, a implementação de uma VPN exige alterações nos *hosts* envolvidos. O usuário enxerga como se estivesse conectado diretamente a sua rede privada, embora o serviço realmente use a infra-estrutura pública para implementar a conexão [24]. Em termos práticos, esta tecnologia permite que uma organização estenda seus serviços de rede sobre a Internet para usuários remotos, filiais e companhias associadas.

As motivações para uso de VPN's são inúmeras, mas podemos dizer que economia e comunicação segura são as principais. As VPNs devem prover quatro pontos críticos para garantir a segurança dos dados [29]:

- Autenticação – assegurar que os dados se originam da mesma fonte que alegam
- Controle de acesso – restringir usuários não autorizados de acessar a rede
- Confidencialidade – impedir qualquer um de ler ou copiar dados que trafegam pela rede
- Integridade – assegurar que ninguém adultere os dados que trafegam pela rede.

Para conferir estas características à VPN, foram desenvolvidas diversas tecnologias de conectividade e segurança. Existem muitas maneiras de se definir o que é uma VPN; descrever uma VPN baseando-se em seus componentes pode ser muito difícil, uma vez que esta pode ser criada de diversas formas.

Uma VPN é constituída de equipamentos, software, protocolos de comunicação e algoritmos de criptografia, integridade e autenticação. Também é constituída de conexões, usuários e políticas de segurança, para realizar a tarefa de criar uma comunicação segura sobre uma rede pública.

Talvez a melhor forma de descrever uma VPN seja defini-la a partir de sua função: estabelecer uma rede privada e confiável de dados entre dois ou mais pontos, sobre uma rede pública não confiável, de baixo custo operacional.

A Internet é uma rede pública e é considerada insegura, pois todas as informações estão sujeitas a interceptação. A VPN vem de encontro a essa necessidade de segurança dentro da rede. Ela permite a interligação das redes locais de uma empresa que estão em regiões geográficas diferentes, utiliza a própria rede pública para fazer a interligação, mas o canal é seguro e ajustável as necessidades da empresa em questão. Na Figura 5.4 tem-se um exemplo de uma VPN que pode interligar redes que estão fisicamente em cidades distintas.

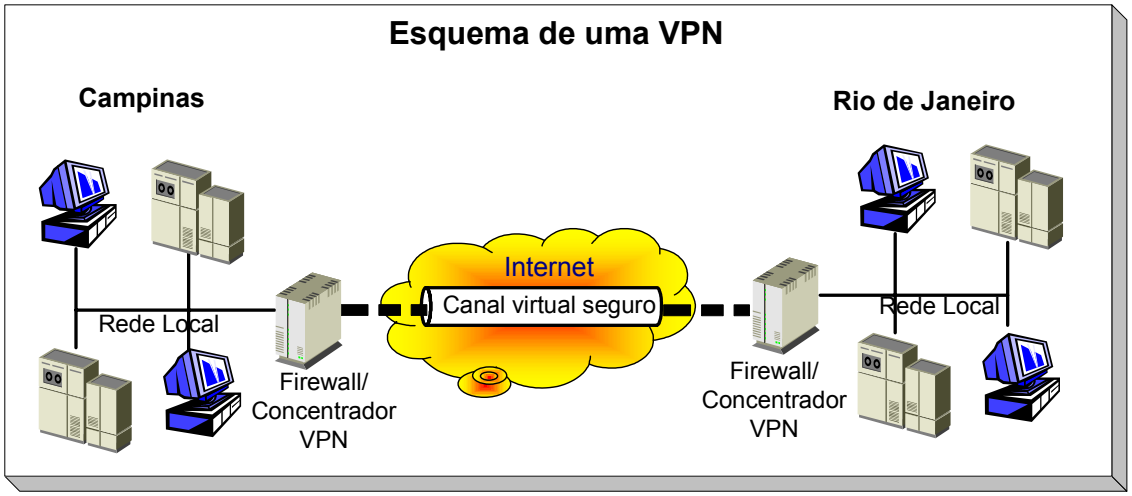

Figura 5.4 – Diagrama esquemático de uma possível configuração VPN

5.2.1.1 Tipos de VPN

As VPNs podem ser classificadas de acordo com diversos critérios: segurança, topologia e tecnologia são alguns deles. Quanto a segurança, as VPN's podem ser confiadas, seguras ou híbridas.

VPN Confiada

Antes de a Internet ser largamente utilizada para tráfego de dados pelas empresas, as VPN's eram constituídas de um ou mais circuitos alugados de um provedor de serviços de dados. Cada circuito funcionava como parte da rede controlada pelo cliente, o que permitia que eles tivessem suas próprias políticas de administração de rede, como segurança e endereçamento IP.

Nessa arquitetura, o cliente confiava ao provedor de acesso a integridade e confidencialidade dos dados. O provedor de serviços, por sua vez, deveria garantir que o acesso aos circuitos era restrito apenas aos clientes que os alugavam.

VPN Segura

Com a popularização da Internet como meio de comunicação de dados, a segurança se tornou uma preocupação muito maior. Por definição, a Internet não tem dono, nem canais ou circuitos que pertençam apenas a um provedor de serviços, o que complica a utilização da VPN confiada. Assim, os fabricantes de equipamentos de telecomunicações começaram a trabalhar em protocolos que permitissem a encriptação e decriptação dos dados nas pontas, independente de sua transmissão pela Internet. A rede transmitiria o dados como outro qualquer.

Esses dados funcionam como um túnel entre as duas pontas: mesmo que um intruso tenha acesso a eles durante a transmissão, não poderá lê-los e qualquer modificação fraudulenta nos dados será detectada na recepção.

VPN Híbrida

Em geral, os mesmos provedores que vendiam serviços de VPN's confiadas, vendem também acesso à Internet. A perda de receita com a migração dos clientes para VPN's seguras fez com que os provedores passassem a oferecer serviços de VPN's seguras sobre a Internet. Nesses casos, a responsabilidade pela segurança dos dados é dividida entre o cliente, que deve adotar políticas de segurança (senhas, por exemplo), e o provedor, que deve administrar corretamente seus equipamentos e sistemas de segurança.

O cliente confia a segurança de seus dados ao provedor, mas o provedor não tem acesso aos mesmos. As VPN's híbridas recebem esse nome por apresentarem características tanto de VPN's confiadas quanto de VPN's seguras.

Quanto a topologia, as VPN's podem ser *LAN-to-LAN*, *Client-to-LAN* ou mistas.

LAN-to-LAN

VPN's com topologia *LAN-to-LAN* interligam duas ou mais *LAN's*[1] através de túneis criados sobre uma rede pública de dados. São como rotas seguras estabelecidas entre redes locais, e podem ser utilizadas, por exemplo, em substituição a uma *WAN*[2], embora sem as restrições geográficas impostas por esta; as redes locais não precisam estar localizadas na mesma cidade, ou no mesmo país, para serem interligadas por uma VPN conforme mostrados na Figura 5.4.

[1] LAN: *Local Area Network*
[2] WAN: *Wide Area Network*

Client-to-LAN

Nessa topologia, clientes remotos individuais utilizam túneis VPN para se conectarem a rede corporativa. Pode ser usada, por exemplo, por técnicos, executivos ou vendedores em campo, que precisem de uma conexão com a rede da empresa independentemente de sua localização.

Mistas

Nessa topologia, redes locais podem ser interligadas tanto a outras *LAN's* quanto a clientes remotos.

5.2.1.2 Componentes de uma VPN baseada em Internet

São quatro os componentes de uma rede privada baseada em Internet: a própria Internet, segurança dos *gateways*, política de segurança dos servidores e certificados de autoridades.

Internet

A Internet providencia o meio de transmissão. Os *gateways* seguros são posicionados entre as redes públicas e privadas e são eles: roteadores, *firewalls*, *hardware* para VPN ou *software* para VPN.

Roteadores

Os roteadores examinam e processam todos os pacotes que saem da LAN, o que os torna candidatos naturais para fazer a criptografia dos pacotes. Os vendedores de serviços de VPN baseados em roteadores normalmente oferecem duas soluções: ou um componente de *software* ou uma placa de circuitos adicional, com um co-processador para criptografia. Esta última é recomendada para situações que requerem grande fluxo de dados.

A utilização de roteadores para criptografia pode ajudar a manter os custos de uma VPN baixos, se estes equipamentos fazem parte da rede já instalada, entretanto, pode aumentar a severidade dos *downtimes* – se o roteador cair, a VPN também cai.

Firewalls

Muitos fabricantes de *firewalls* incluem a capacidade de tunelamento em seus produtos. Como os roteadores, os *firewalls* também precisam processar todo o tráfego IP, e decidir quais pacotes serão aceitos ou barrados. Essa tarefa já exige muito processamento do *firewall*, que, portanto não é o melhor elemento para fazer o tunelamento em VPN's de grande capacidade.

A utilização de *firewalls* para fazer VPN's pode ser interessante, principalmente em redes de pequena capacidade, pois diminui custos operacionais e com equipamento, assim como nos roteadores, quando a VPN é feita no *firewall,* uma falha neste último irá afetar também à VPN.

***Hardware* para VPN**

Outra solução para VPN é a utilização de *hardware* específico, desenhado para fazer tunelamento, criptopgrafia e autenticação dos dados. Esses equipamentos são normalmente inseridos entre os roteadores e os *links* das *WAN's* e, apesar de a maioria desses produtos ser destinada a VPN's *LAN-to-LAN*, alguns suportam também conexões de clientes.

A utilização de *hardware* especializado apresenta algumas vantagens sobre as outras soluções: em primeiro lugar, elimina-se o ponto único de falha na rede, caso haja algum problema com o equipamento. Outro ponto é o alto desempenho, e a possibilidade de integração de outros serviços ao *gateway* de segurança, como gerenciamento de banda e priorização de tráfego na VPN.

Servidor de políticas de segurança

Além do *gateway* de segurança, outro componente importante de uma VPN é o servidor de políticas de segurança. Esse servidor mantém as listas de controle de acesso e outras informações relacionadas aos usuários que o *gateway* de segurança usa para determinar qual tráfego é autorizado.

Autoridades Certificadoras

Autoridades certificadoras são necessárias para verificar a autenticidade das chaves compartilhadas entre os *sites* e podem também ser usadas para verificar indivíduos, através da utilização de certificados digitais. As instituições/empresas podem utilizar um servidor corporativo de certificados digitais, deste modo ter seu próprio banco de dados de certificados digitais.

Se a VPN também é utilizada como *Extranet*, pode ser necessária a utilização de entidades certificadoras externas, para verificar a identidade dos parceiros comerciais que acessam a rede.

5.2.2 Relações de Confiança (*Trust Relationship*)

Relação de confiança é o nome que se dá a técnica que proporciona ao usuário, devidamente autenticado em um servidor (confiável), ter acesso ao um outro servidor (confiante) sem que esta autenticação seja novamente necessária através de uma conta específica ou um novo *login*, pois este confia na autenticação realizada pelo confiável [16].

À primeira vista parece um pouco confuso, mas a idéia é simples e aplicada em diversos tipos de ambientes computacionais, principalmente em servidores UNIX e *Windows*, cada qual com sua particularidade para implementação.

É uma técnica que precisa ser realizada com muito cuidado, conforme pode-se perceber, pois se um atacante conseguir, com sucesso, uma autenticação em um servidor vulnerável, terá acesso a todos

os servidores com os quais possui relação de confiança, e também as recursos disponíveis deste(s) servidor(es).

A diferença entre o modelo de relacionamento de confiança em servidores Unix e *Windows* está no fato que no *Windows NT Server* os relacionamentos de confiança são estabelecidos entre domínios, já no Unix podem ser estabelecidos entre *hosts* [19].

Cada relacionamento de confiança é configurado como uma relação unidirecional, não recíproca e não transitiva. Isto significa que o fato de um domínio "A" estar configurado para confiar em um domínio "B" não implica que o domínio "B" irá confiar no domínio "A". Tal condição exigiria a configuração de um segundo relacionamento de confiança, no qual fossem invertidos, isto é, o domínio "B" confiasse explicitamente no domínio "A".

Da mesma forma, o relacionamento de confiança estabelecido entre dois domínios não é estendido a outros domínios pelo fato de haverem relacionamentos de confiança adicionais. Assim se o domínio "A" confia em "B" e supomos que "B" confie em um terceiro domínio "C", não será estabelecido com isso que "A" confia em "C". Isso só aconteceria se fosse criado um relacionamento de confiança entre os domínios "A" e "C".

Uma vantagem do relacionamento de confiança é que podem simplificar o gerenciamento de contas e direitos de acesso em redes complexas, proporcionando uma estrutura mais escalável e possibilitado o emprego de métodos de administração distribuída [19].

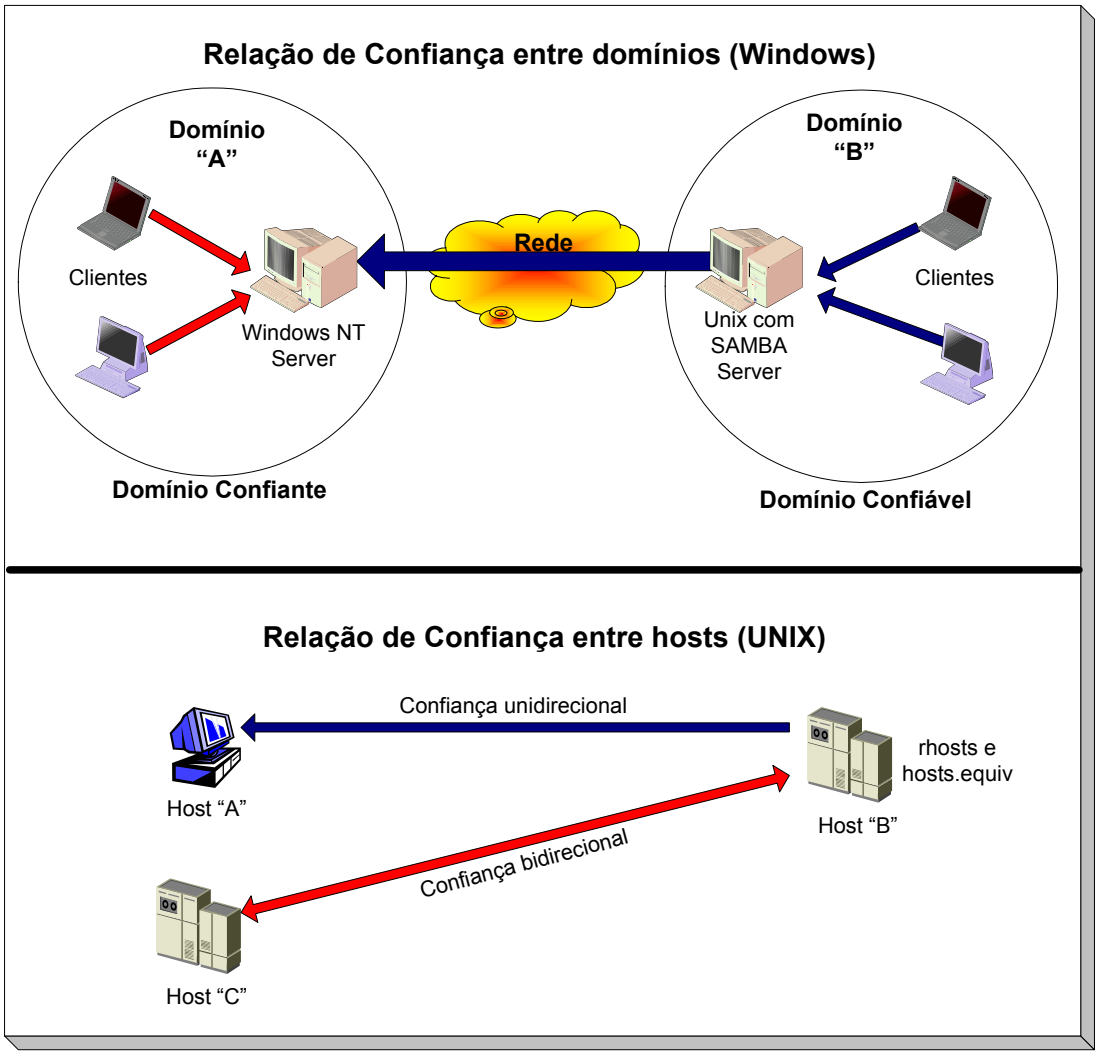

Figura 5.5 – Esquema de Relação de Confiança em ambientes *Windows* e Unix

Na Figura 5.5 mostra que é possível interligar domínios utilizando servidores Unix com o *SAMBA Server* instalado com controladores de domínio *Windows NT Server*. Já para se criar uma relação de confiança entre *hosts* UNIX é necessário alterar alguns arquivos do sistema operacional como os arquivos ***rhosts*** e ***hosts.equiv***.

Com o surgimento da família *Windows 2000 Server* também surgiu um novo modelo da *Microsoft* para administração do domínio e seus recursos denominado *Active Directory*. O *Active Directory* representa uma iniciativa para estabelecer um gerenciamento integrado de recursos, distinto e aperfeiçoado em relação àquele oferecido pelo *Windows NT*, que dependia de relação de confiança para

integração entre domínios. Com o *Active Directory* os relacionamentos de confiança são substituídos por uma representação unificada do conjunto de recursos disponíveis em vários domínios [19].

5.2.3 Certificação de Clientes e Servidores

De uma maneira simples, o Certificado Digital é a versão eletrônica da sua identificação de usuário na rede (usuário e senha). O Certificado Digital é como se fosse a "carteira de identidade" do usuário na rede. No *Windows 2000/2003 Server*, o certificado digital do usuário também é conhecido (na documentação oficial), como um Certificado de chave pública, uma vez que uma das informações gravadas no certificado digital do usuário é justamente a sua chave pública [04].

Um certificado de chave pública, geralmente chamado somente de certificado, é uma declaração assinada digitalmente que vincula o valor de uma chave pública a identidade da pessoa que pode ser a conta do usuário no Diretório Ativo, dispositivo ou serviço que contém a chave privada correspondente.

Certificados podem ser emitidos para uma série de funções, como autenticação de usuário na Internet, autenticação de um servidor *web*, correio eletrônico seguro (S/MIME), *IPSec*, para utilização com o protocolo TLS e assinatura de códigos. Os certificados digitais tem que ser emitidos por uma Autoridade Certificadora (*Certificate Authority*).

Com o *Windows 2000/2003 Server* está disponível o *Microsoft Certificate Services*, que é um servidor que permite criar uma autoridade certificadora na própria rede da empresa, sem ter que fazer uso de uma entidade certificadora externa. Ao utilizar o *Certificate Services* para a emissão e gerenciamento de certificados, os certificados digitais poderão ser utilizados pelos usuários para fazer o *logon* na rede. Os certificados também são emitidos de uma autoridade de certificação para outra a fim de estabelecer uma hierarquia de certificação [04].

A maioria dos certificados em uso hoje atualmente são baseados no padrão X.509. Esta é a tecnologia fundamental usada na *Public Key Infrastructure* (PKI).

Normalmente, os certificados contêm as seguintes informações:

- Chave pública do usuário
- Informações da identificação do usuário (como o nome e o endereço de correio eletrônico)
- Período de validade (o período de tempo em que o certificado é considerado válido)
- Informações sobre a identificação do emissor do certificado.
- A assinatura digital do emissor, que atesta a validade da ligação entre a chave pública do usuário e as informações de identificação do usuário.

Um certificado só é válido pelo período de tempo nele especificado, ou seja, o certificado tem prazo de validade e tem que ser renovado periodicamente. Esta é uma medida importante para garantir aumentar o nível de segurança, pois a cada renovação, um novo par de chaves é gerado [04].

Cada certificado contém datas "Válido de" e "Válido até", que limitam o período de validade. Depois que o período de validade de um certificado terminar, um novo certificado deve ser solicitado pelo usuário do agora expirado certificado.

Em situações em que seja necessário desabilitar um certificado, este pode ser revogado pelo emissor. Cada emissor mantém uma lista de certificados revogados (CRL – *Certification Revocation List*), a qual é usada pelos programas quando a validade de um determinado certificado está sendo verificada. Por exemplo, programas que usam certificados para autenticação, ao receberem uma tentativa de acesso, primeiro entram em contato com a autoridade certificadora (no caso do *Windows 2003 Server* um servidor com o *Microsoft Certificate Service*) para verificar se o certificado que está sendo apresentado para *logon*, não está na lista dos certificados revogados – CRL. Se o certificado estiver na CRL, o *logon* será negado.

5.2.3.1 Certificados e Autoridades de Certificação

Todo certificado é emitido por uma Autoridade de Certificação. A autoridade de certificação, a partir de agora denominada apenas CA, é responsável pela verificação sobre a veracidade dos dados do usuário que está requisitando o certificado [04].

Uma autoridade de certificação é uma entidade encarregada de emitir certificados para indivíduos, computadores ou organizações, sendo que os certificados é que confirmam a identidade e outros atributos do usuário do certificado, para outras entidades. Ela aceita uma solicitação de certificado, verifica as informações do solicitador e, em seguida, usa sua chave privada para aplicar a assinatura digital no certificado, caso aprovado, emite então o certificado para que o usuário do certificado o use como uma credencial de segurança dentro de uma infra-estrutura de chave pública (PKI).

Uma autoridade de certificação também é responsável por revogar certificados e publicar uma lista de certificados revogados (CRL), ela pode ser uma empresa que presta o serviço de autoridade certificadora ou ser uma autoridade de certificação criada para ser usada pela própria organização, instalando os Serviços de certificados.

Cada autoridade de certificação pode ter requisitos diferentes de prova de identidade, como uma conta de domínio do Diretório Ativo, crachá de empregado, carteira de motorista, solicitação autenticada ou endereço físico dentre outros. Verificações de identificação como essa geralmente asseguram uma autoridade de certificação no local, de tal modo que as organizações possam validar seus próprios empregados ou membros.

As autoridades de certificação corporativas, como do *Windows 2000/2003 Server,* usam as credenciais da conta de usuário do Diretório Ativo de uma pessoa, como prova de identidade. Neste exemplo, se um usuário tiver efetuado *logon* em um domínio do *Windows 2000/2003 Server* e solicitar um certificado de uma autoridade de certificação corporativa, a autoridade de certificação saberá é este usuário através do Diretório Ativo, que vai dizer "quem ele é".

Todas as autoridades de certificação têm um certificado para confirmar sua própria identidade, emitido por outra autoridade de certificação confiável ou, no caso de autoridades de certificação raiz, emitido por elas mesmas. É importante lembrar que qualquer pessoa pode criar uma autoridade de certificação, a questão real é se você, como um usuário ou um administrador, confia naquela autoridade de certificação e, por extensão, nas diretivas e procedimentos que ela emprega para confirmar a identidade dos certificados emitidos para entidades por essa autoridade de certificação.

Em uma rede baseada no *Windows 2000/2003 Server*, o administrador também pode utilizar uma CA externa. Porém, com o uso do *Microsoft Certificate Services*, o administrador pode criar sua própria autoridade certificadora. Este serviço permite a criação de sofisticados ambientes de certificação, com a criação de uma hierarquia de CA's. Com o uso do *Certificate Services* podem ser criadas os seguintes tipos de autoridades certificadoras:

- *Enterprise Root CA.*
- *Enterprise Subordinate CA.*
- *Standalone Root CA.*
- *Standalone Subordinate CA*

Enterprise Root CA – Autoridade certificadora corporativa raiz: Um único servidor pode ser configurado como *Enterprise root CA* em uma floresta de domínios.

Este servidor ocupa o topo da hierarquia de autoridades certificadoras. Normalmente não é utilizado para emitir certificados para usuários ou computadores, mas sim para autoridades certificadoras corporativas subordinadas. Os certificados para usuários e computadores são emitidos

pelas autoridades subordinadas. Com isso pode-se criar uma hierarquia de autoridades certificadoras, de tal maneira que a emissão de certificados seja efetuada por um servidor do próprio domínio do usuário.

Outro detalhe importante é que a autoridade certificadora raiz é responsável por assinar o seu próprio certificado (afinal não há nenhuma autoridade acima dela). Isso é que caracteriza esta autoridade como uma autoridade certificadora raiz.

Enterprise Subordinate CA – Autoridade certificadora Corporativa subordinada: Para instalar uma autoridade certificadora corporativa subordinada, é necessário ter acesso ao certificado da autoridade certificadora corporativa raiz.

O uso deste certificado é que liga a autoridade certificadora que está sendo instalada, com uma autoridade subordinada a autoridade certificadora raiz, formando uma hierarquia de entidades certificadoras. Este tipo de autoridade pode emitir certificados para usuários e computadores do Diretório Ativo ou para outras autoridades certificadores subordinadas de níveis mais baixos, aumentando desta maneira, o número de níveis da hierarquia de autoridades certificadoras.

Stand-alone Root CA – Autoridade certificadora autônoma raiz: Este tipo de autoridade certificadora não depende do Diretório Ativo. Pode ser utilizado, por exemplo, para emitir certificados para parceiros de negócio e prestadores de serviço, que precisam de certificados digitais para acessar determinadas áreas da Intranet ou da Extranet da instituição.

Uma vantagem adicional é que um servidor configurado como autoridade certificadora autônoma raiz, pode ser desconectado da rede, como uma garantia adicional de segurança. Este tipo de autoridade certificadora também é responsável por emitir os certificados de registro das autoridades certificadoras autônomas subordinadas.

Stand-alone Subordinate CA - Autoridade Certificadora Autônoma Subordinada: Este tipo de autoridade certificadora está subordinada a uma autoridade certificadora autônoma raiz.

O processo normalmente é o mesmo utilizado para o caso das autoridades certificadoras corporativas, ou seja, a autoridade certificadora autônoma raiz não é utilizada para emissão de certificados para usuários e computadores, mas sim para a emissão de certificados para as autoridades certificadoras autônomas subordinadas. As autoridades certificadoras autônomas subordinadas é que são responsáveis pela emissão dos certificados para usuários e computadores.

A existência de uma autoridade certificadora significa que você tem confiança de que a autoridade de certificação possui as diretivas corretas no local correto e ao avaliar as solicitações de certificado, irá negar certificados para qualquer entidade que não atender a essas diretivas. Esta é uma questão fundamental para garantir a identidade dos usuários.

Ao fazer uma verificação rigorosa dos dados informados, antes de emitir um certificado para um usuário, servidor ou computador, a CA garante que quem obtém o certificado realmente é quem diz ser – prova de identidade. Por isso a importância fundamental de definir uma metodologia clara, simples e de fácil execução, para a verificação dos dados, antes de emitir os certificados [04].

Para serviços, computadores e usuários do *Windows 2000/2003 Server*, a confiança em uma autoridade de certificação é estabelecida quando você possui uma cópia do certificado raiz no armazenamento das autoridades de certificação raiz confiáveis e tem um caminho de certificação válido, significando que nenhum dos certificados no caminho de certificação foi revogado ou que seus períodos de validade expiraram.

O caminho de certificação inclui todos os certificados emitidos para cada CA na hierarquia da certificação de uma CA subordinada para a CA raiz [04].

Conclusão

Após o surgimento dos protocolos seguros, o próximo passo seria dado se fosse encontrada uma maneira de considerar os canais públicos ou abertos como um caminho de rede também seguro. E as instituições de pesquisa e esforços pessoais conseguiram uma maneira para solução deste problema. Como principal resultado destas pesquisas e a que melhor representa este esforço é a criação da tecnologia de VPN's.

Podemos dizer que hoje, o leque de opções disponíveis para implantação de uma aplicação Cliente/Servidor é muito maior, independente do tipo de plataforma desejada, pois grande parte da solução encontra-se disponível como software livre e de código aberto. Somando-se a isso temos o fato que podemos agregar diversas técnicas para alcançar este objetivo.

A exposição de um estudo de caso real, através de uma aplicação Cliente/Servidor proprietária, expõe as principais características sobre como são desenvolvidas as aplicações comerciais, utilizadas principalmente na plataforma *MS-Windows*.

O conhecimento destas características pode oferecer subsídios suficientes, para que os administradores de sistemas possam tomar decisões importantes no que tange aos mecanismos necessários para implantação de produtos de *software* de forma integrada, transparente ao usuário e segura.

Em resumo, hoje existem métodos, tecnologias e condições para se cumprir todos os requisitos que regem uma comunicação segura como disponibilidade, integridade, confidencialidade, autenticidade, não repúdio e controle de acesso aos recursos.

Capítulo 6

Propostas de soluções para o estudo de caso

A proposta deste Capítulo é apresentar as diversas formas para solucionar os problemas relativos a segurança, escalabilidade e compatibilidade para a execução da aplicação Cliente/Servidor, mostrada anteriormente em nosso caso de estudo no Capitulo 5.

Os conceitos e métodos seguros apresentados nos capítulos 4 e 5 foram colocados em prática. Algumas soluções não puderam ser adequadas devido as particularidades da aplicação, outras se mostraram insuficientes ou de difícil administração e, desta maneira, prejudicavam a utilização da aplicação mesmo em ambientes homogêneos (soluções totalmente proprietárias ou *software* livre).

Ao longo do desenvolvimento deste Capítulo, serão apresentadas as propostas implementadas e analisadas, com a exposição da solução adotada justificando-se suas razões, bem como os fatores que levaram a eliminação de algumas propostas.

6.1 Análise das soluções viáveis

Devido as características da aplicação e seu modelo de construção, que não permite uma solução completa com a utilização de *software* livre, foi necessária uma série de pesquisas e testes práticos. O objetivo foi avaliar, dentre as possíveis soluções, a que melhor pode ser aplicada, levando-se em consideração aspectos fundamentais para prover a segurança, a escalabilidade, a compatibilidade e a administração do sistema.

Para compreender como pôde-se chegar a uma solução é necessário rever alguns conceitos básicos sobre autenticidade e segurança na plataforma *MS-Windows*. Deste modo, buscou-se soluções existentes para esta plataforma, sem descartar a possibilidade da utilização de outras.

Ao longo dos últimos 11 anos a *Microsoft* vem desenvolvendo seus sistemas operacionais para ambientes corporativos e servidores de grande porte. Uma prova disso é a evolução da família *Windows NT Server* até a concepção da família 2003 *Server*.

Neste caminho muitas falhas de segurança e *bugs* na construção deste *software* desafiaram a capacidade desta plataforma de garantir a privacidade e a segurança da informação administrada por estes sistemas. Além disso a família *Windows NT* não possui ferramentas nativas capazes de oferecer aos administradores recursos mais avançados para atender a demanda das novas aplicações, muito mais abrangentes e que, em sua maioria, utilizam como protocolo padrão TCP/IP e não mais protocolos proprietários. Somando-se a isso têm o aspecto da mobilidade: a tendência é que os usuários do sistema estejam fisicamente fora de suas empresas/instituições e, portanto não estarão utilizando redes seguras. O acesso pode partir de qualquer parte onde exista um ponto de rede conectado a Internet.

Em vista deste novo cenário, a *Microsoft* incorporou a nova família de Sistemas Operacionais, funcionalidades e ferramentas capazes de aumentar o leque de opções para atender os requisitos de segurança e autenticidade. A partir da versão *MS-Windows 2000* já é possível utilizar protocolos seguros, criar entidades certificadoras, utilizar e administrar informações de um diretório ativo e muito mais. É com base nestas ferramentas novas, que se chegou a solução do problema de implantação da nossa aplicação em estudo, contudo é importante ressaltar algumas características que diferem esta plataforma das demais, incluindo principalmente o processo de autenticação do usuário e sua identificação na rede.

6.1.1 Provendo a autenticidade na plataforma *MS-Windows*

Para entendermos como podemos garantir a autenticidade de um usuário em redes organizadas através de domínios *Windows*, ou mesmo diretório ativo, precisamos saber como é realizado o processo para armazenar e autenticar os dados dos usuários do domínio ou do diretório ativo.

Todo acesso aos recursos do sistema (domínio, diretório ativo ou na máquina local) é baseado nas permissões do usuário. Esta conta pode pertencer a uma série de grupos criados pelo administrador do sistema, diferentemente do que ocorre em sistemas UNIX [22]. Cada conta recebe uma identificação denominada ***SID*** (*Security Identifier*) com 28 bytes, gerenciado e armazenado pelo controlador do

domínio ou diretório ativo. A informação é armazenada no registro da máquina controladora ou mesmo localmente, nos casos em que não está ligada a nenhum domínio ou diretório ativo. Um algoritmo criptográfico é utilizado para codificar a informação, o padrão para servidores *Windows NT* é o protocolo NTLM[1] e para servidores *Windows 2000 Server* é o *kerberos v5*. Desta maneira temos conforme [22]:

Campo	Bytes	Valor	Observação
Revision	1	0x01	*Revision Number*
Count	1	5	
Top Level Auth.	6	0x05	*SECURITY_NT_AUTHORITY*
Subauth.	4	0x15	*SECURITY_NT_NON_UNIQUE*
Subauth.	12	-	Identificador da máquina
Subauth.	4	-	Identificador do usuário

Tabela 6.1 – Distribuição dos 28 bits do SID

Uma cadeia alfanumérica é armazenada no registro como abaixo.

S-1-5-21-1179599015-1994013950-622671684

Sendo assim, quando um usuário faz o procedimento de *logon* na rede existe a interação mostrada na Figura 6.1 entre o cliente e o servidor do domínio ou diretório ativo.

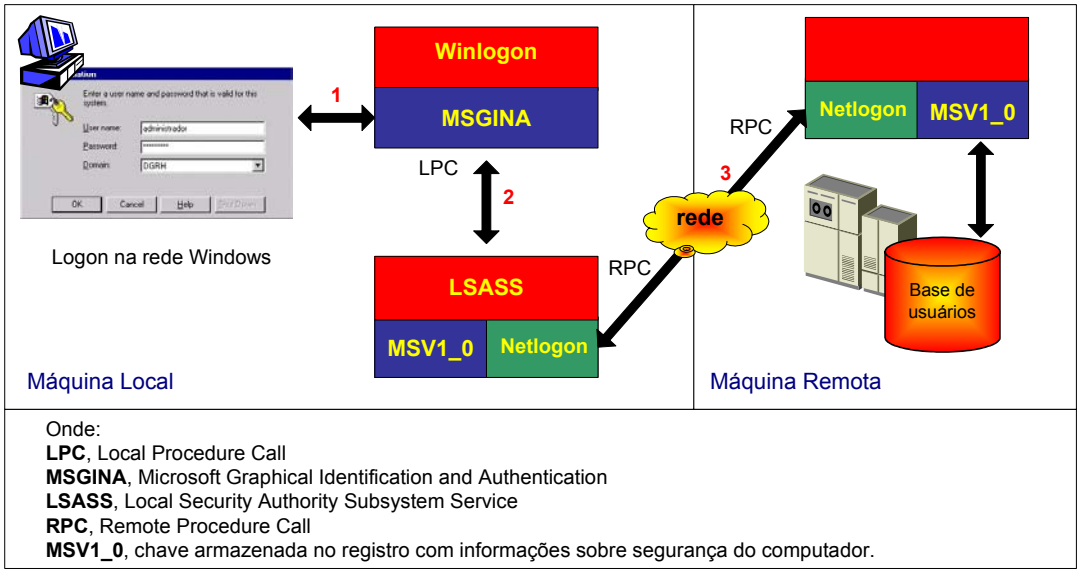

Figura 6.1 – Interação para autenticação de usuários em domínios *Windows*

[1] NTLM: NTLanMan (*Windows NT Lan Manager,* Protocolo de Autenticação)

Se a aplicação estiver em um ambiente integrado a um domínio *Windows* 2000, o esquema de autenticação utilizará novas bibliotecas e novos algoritmos para autenticar a informação, como por exemplo, o protocolo *kerberos* (Capítulo 4, item 4.2.2). A base de dados de usuários, neste caso, passa a ser um objeto do Sistema Operacional.

6.1.2 Utilização de protocolos seguros na plataforma *MS-Windows*

O meio mais eficaz para se garantir a segurança das informações, que são trocadas entre o cliente da aplicação e o servidor, é implementar um protocolo seguro, neste caso o protocolo adequado para esta finalidade é o *IPSec* (item 4.1.6), devido as suas características e sua integração com protocolo IP. Contudo não se pode esquecer que as estações clientes são dotadas de sistemas operacionais desenvolvidos pela *Microsoft,* com suas características e limitações.

Sendo assim temos que levar em consideração todas as versões desenvolvidas para a plataforma IBM-PC de 32 bits que são: *MS-Windows* 95/98/*Millennium* e os derivados da plataforma *Windows NT Server e Workstation*, *Windows 2000 Server/Professional, Windows XP Professional* e *Windows 2003 Server*.

Evidentemente nem todos os Sistemas Operacionais testados podem ser utilizados na solução, pois a implementação do protocolo *IPSec* ocorre apenas nas versões para *desktop* do *Windows 2000* em diante. Entretanto existem no mercado empresas que desenvolvem mecanismos para se implantar o protocolo *IPSec* em estações de trabalho com Sistemas Operacionais mais antigos, porém não de forma nativa.

Através do uso do protocolo *IPSec*, todo o tráfego entre o cliente e o servidor ocorre de forma criptografada, incluindo os comandos utilizados no transporte SMB para acessar o diretório remoto no servidor.

6.2 Análise da solução com a utilização de VPN *Microsoft*

Uma das melhores maneiras para se agregar, ao mesmo tempo, segurança e autenticidade é através do uso da VPN. Porém existe uma diferença fundamental entre o conceito de VPN para a plataforma da *Microsoft* e redes TCP/IP.

A solução de VPN da *Microsoft* é um produto integrado ao Sistema Operacional, que atua na camada de enlace da arquitetura TCP/IP e, desta maneira, os clientes da VPN estabelecem um túnel através da rede utilizando o conjunto de protocolos *L2TP/IPSec* e *PPTP* da seguinte forma [41]:

• *L2TP/IPsec*, utilizado métodos de autenticação PPP em nível de usuário e autenticação *IPSec* em nível de *kernel* para certificados e autenticação de dados, também integridade e criptografia

• *PPTP*, usando método de autenticação *PPTP* em nível de usuário e MPPE (*Microsoft Point-to-Point Encryption*) para criptografia de dados.

Devido a estas características o uso da VPN torna-se limitado. Tipicamente os clientes da VPN são de acordo com [41]:

• Usuários com laptop que precisam conectar-se a Intranet da organização para acessar *e-mail* e outros recursos através de redes externas ou inseguras.

• Pessoas que precisam utilizar a Internet para acessar os recursos da organização através de redes domésticas.

• Administradores remotos que utilizam a Internet para conectar-se a uma rede da organização e configurar redes, serviços e aplicações.

Para se permitir o uso de aplicações Cliente/Servidor em ambientes de VPN, todos os elementos envolvidos na comunicação devem fazer parte deste cenário, ou seja, possuir endereços de rede distribuídos pelo concentrador da VPN, conforme mostrado na Figura 6.2, pois o acesso aos recursos do servidor pelo cliente é verificado através do seu SID.

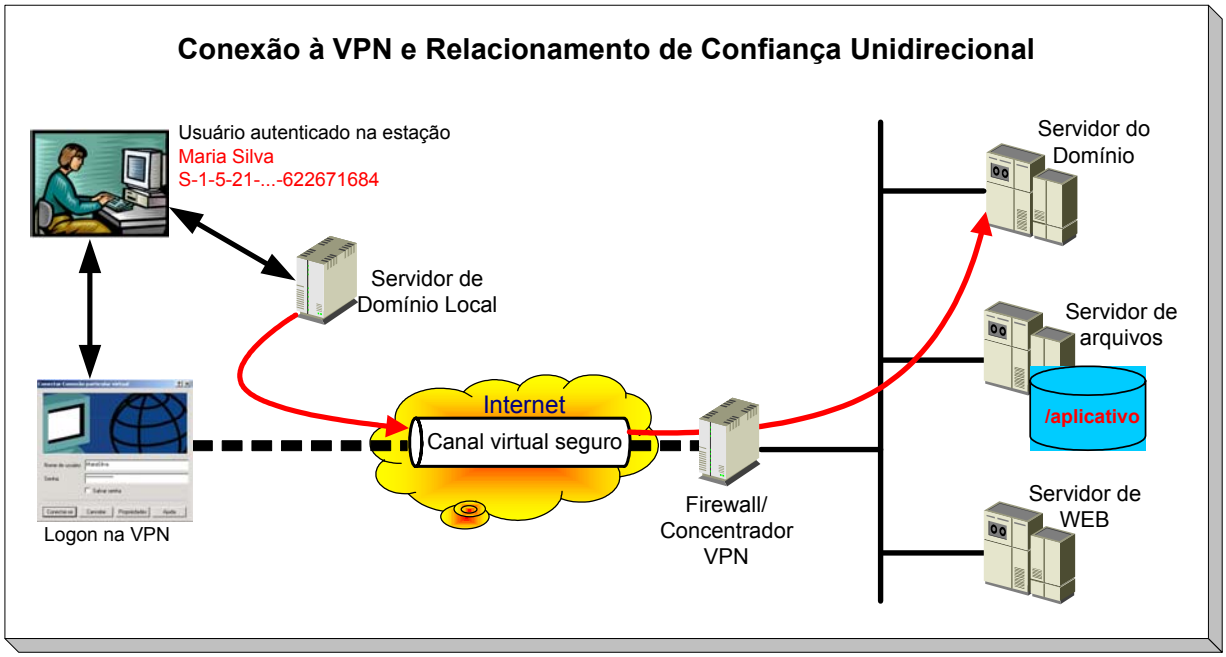

Figura 6.2 – Cenário da Solução utilizando VPN *Microsoft*

Do ponto de vista da segurança e da autenticidade o problema é todo resolvido. Comprovadamente os protocolos e métodos utilizados numa VPN atendem perfeitamente, entretanto existe a questão da administração deste ambiente e também a escalabilidade. A adição de novos usuários ou grupos de usuários que podem acessar o recurso de disco "/Aplicativo" no servidor de arquivos, não será problema, desde que o servidor de domínio/diretório ativo esteja integrado a VPN.

Faixas de endereços destinados a VPN devem ser reservados e conhecidos dos administradores, pois em caso de perda da conexão, o servidor local pode ser autenticado novamente na VPN com seu endereço de rede conhecido. Outro cuidado que se deve tomar é com redes locais que utilizam classes reservadas de endereços da arquitetura TCP/IP através de NAT (*Network Address Translation*), uma vez que a maioria das VPN's são formadas por endereços de classe semelhante, o que não impede o uso de endereços reais, quando for o caso.

Em virtude destes fatos, a administração desta solução torna-se algo complexo a medida em que novos clientes e servidores são adicionados. A questão da escalabilidade está intimamente ligada a questão da quantidade de clientes que utilizarão o Sistema, evidentemente existe um limite tanto de *hardware* quanto de *software* que deve ser levado em consideração na montagem desta solução. O elevado número de usuários e conexões podem degradar a utilização da aplicação [29].

6.3 Análise da solução VPN com *IPSec* nativo da plataforma *Microsoft*

A segunda alternativa para a solução do problema é a utilização do protocolo seguro *IPSec* de forma nativa na plataforma *MS-Windows,* criando uma VPN de *facto*, utilizando ferramentas disponíveis na plataforma onde é executada a aplicação, observando-se em garantir que todos os princípios que regem a comunicação (item 3.2.4) sejam cumpridos, independente do caminho que a informação percorre de uma origem ao seu destino.

Tecnicamente esta é uma solução simples de ser adotada em aplicações do tipo cliente/servidor desenvolvidas com as mesmas características e finalidades do nosso estudo de caso. Utiliza-se a combinação de duas técnicas: Relação de Confiança entre os servidores de domínio/diretório ativo e utilização do *IPSec* como protocolo de comunicação.

O principal problema encontrado é garantir que apenas usuários credenciados tenham acesso a um diretório compartilhado em um servidor corporativo, em muitos casos não apenas para leitura, mas também para escrita, além disso, é necessário compreender o limite imposto para a instalação nos

clientes, pois aceita apenas estações com Sistema Operacional padrão *MS-Windows*, devido as características de construção da aplicação (na versão cliente/servidor).

A Figura 6.3 demonstra o primeiro passo da solução, que é a autenticação do usuário por um servidor de domínios *Windows NT/Samba Server* ou *Windows 2000/2003 Server* (Diretório Ativo).

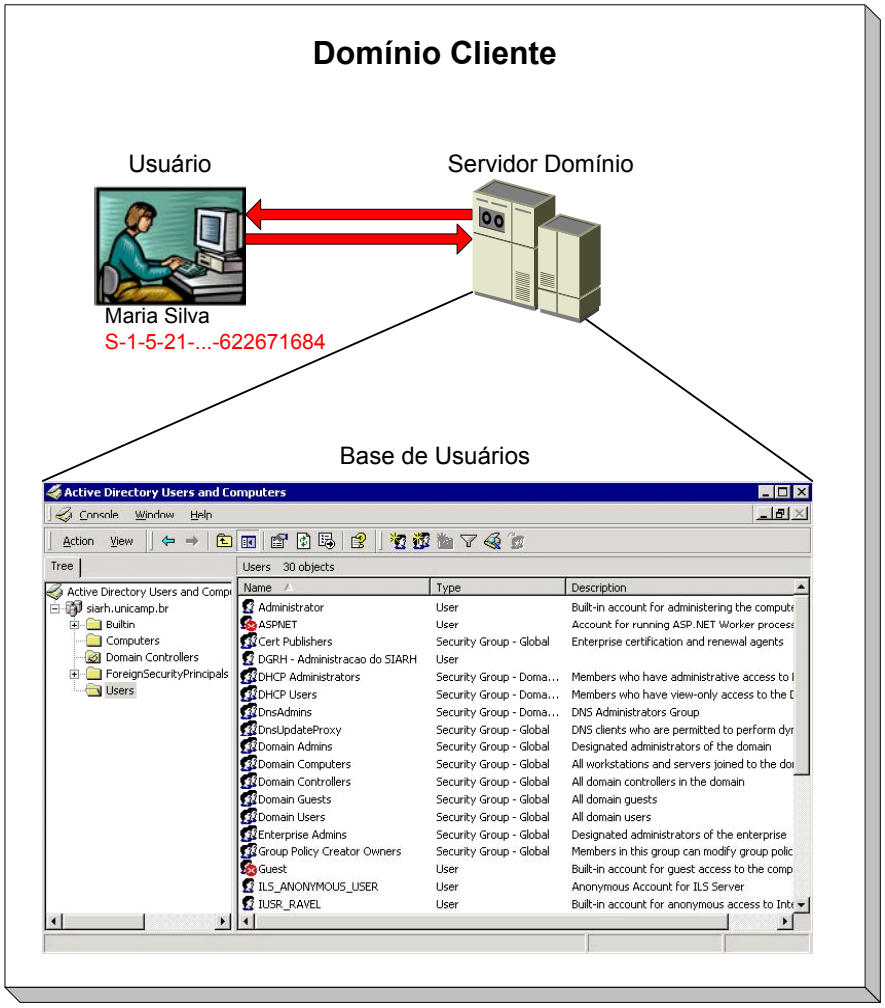

Figura 6.3 – Autenticação do usuário em um domínio local *Windows*

Neste exemplo o usuário "**Maria Silva**" está se autenticando em um servidor que controla um Diretório Ativo, portanto ela possui uma conta neste domínio que a identifica unicamente. Nesse tipo de ambiente o usuário deve pertencer (no mínimo) ao grupo de usuários do sistema, porém o Administrador pode criar uma série de outros grupos, agrupando os usuários de acordo com sua política de administração, inclusive recebendo novos usuários de domínios diferentes, em que possa confiar. Esse é o próximo passo, ou seja, criar uma relação de confiança entre domínios.

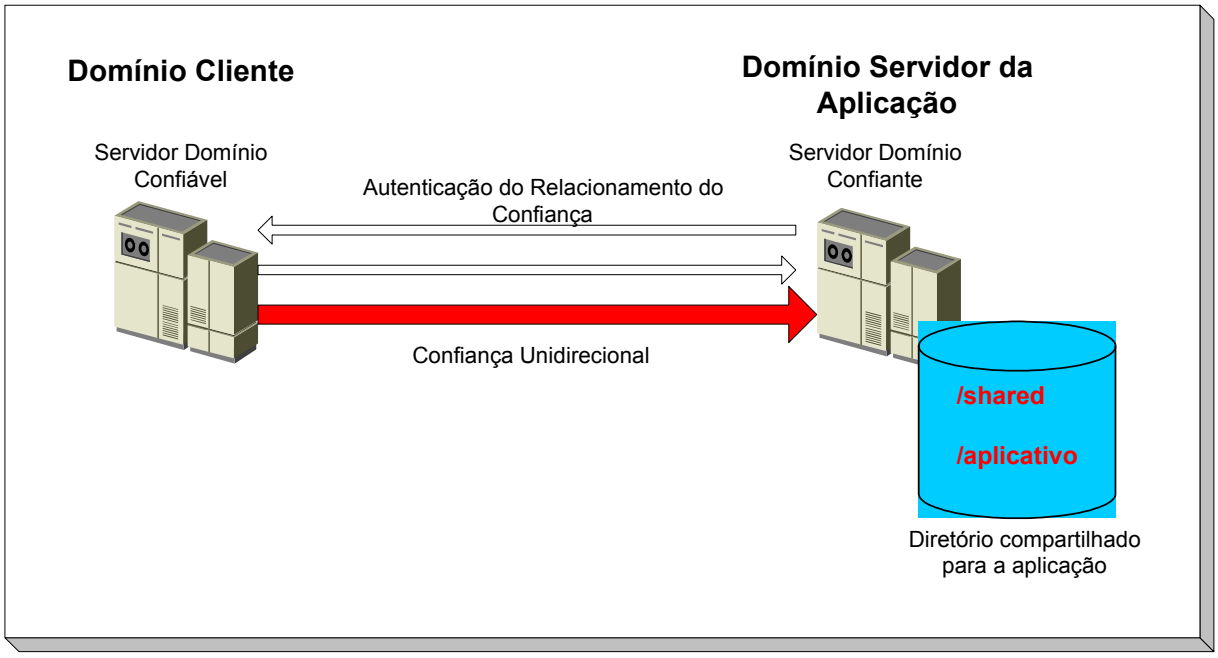

Figura 6.4 – Relacionamento de Confiança para acesso ao diretório compartilhado por usuários da aplicação

Através da relação de confiança unidirecional, mostrada na Figura 6.4, o administrador do domínio confiante poderá escolher quais usuários ou grupos do domínio confiável terão acesso ao diretório compartilhado, bem como definir as permissões de acesso destes usuários e métodos de auditoria. Deste modo cria-se uma rede de servidores confiáveis com a administração de usuários de forma descentralizada, ao contrário das VPN's que necessitam do concentrador de VPN e dos usuários cadastrados no servidor. Vale lembrar que as conexões entre as estações clientes e o servidor utilizam o protocolo *IPSec* nas comunicações.

A Figura 6.5 apresenta o esquema utilizado para disponibilizar o diretório **"/aplicativo"** para domínios clientes, neste diretório estão instalados os arquivos da versão *SERVER* da aplicação, além dos programas que possibilitam a instalação remota da versão CLIENTE nas estações. Pode-se encontrar também os arquivos que são gerados pelos clientes e armazenados temporariamente no servidor como relatórios, gráficos, documentos e imagens.

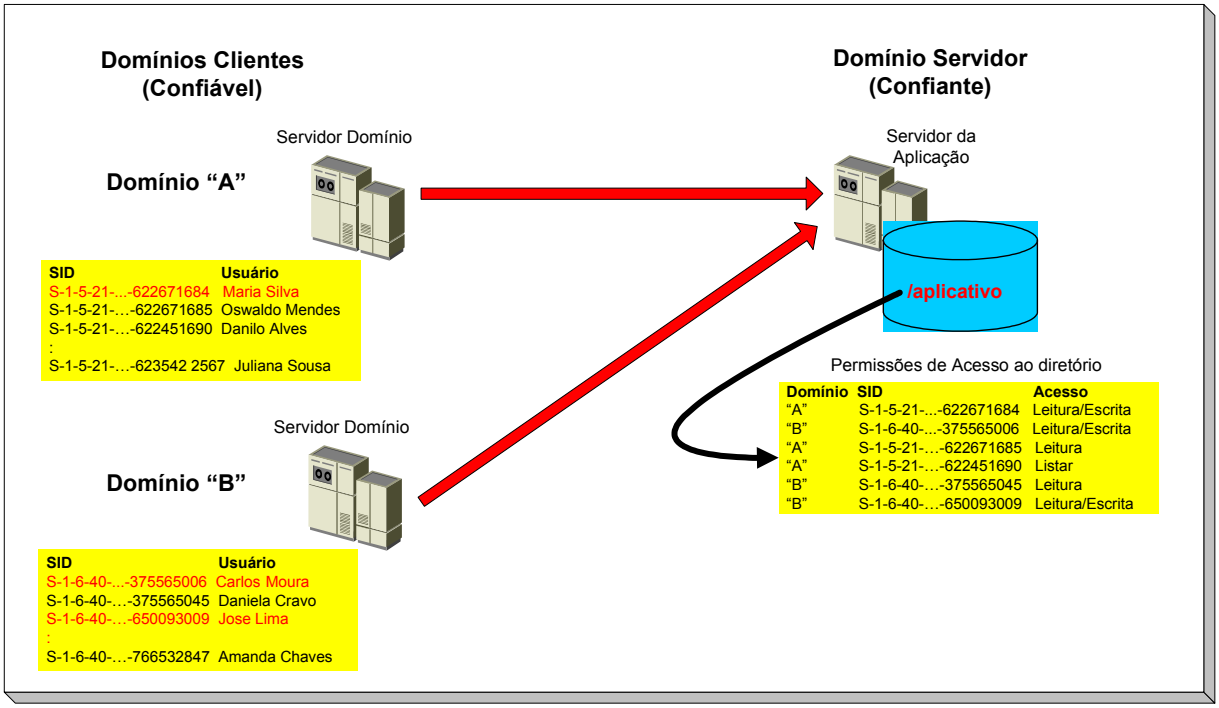

Figura 6.5 – Definindo usuários e permissões de acesso ao diretório compartilhado

Uma vez garantida a identidade dos usuários, não se deve esquecer que estas aplicações utilizam bancos de dados clientes, neste caso o banco *Oracle* 9i. Sendo assim é necessário definir parâmetros na BDE que possibilitem a comunicação nativa entre o servidor do banco de dados e o cliente, além de alterar parâmetros na configuração cliente de forma a habilitar autenticação e criptografia.

Por sua vez o administrador do banco de dados deve possuir regras que protejam a base de dados. Isso inclui controle de acesso, integridade dos dados, criptografia e auditoria. O banco *Oracle* possui suporte para uma série de algoritmos criptográficos como:

- *RSA Data Security RC4*: este algoritmo permite a criptografia dos dados de forma rápida, utilizando uma chave privada gerada randomicamente em cada sessão com o banco. Todo o tráfego ocorre de maneira segura, incluindo os comandos SQL, os valores associados a estes comandos, resultados e chamadas de procedimentos (*stored procedure call*). O cliente, servidor ou ambos, podem requisitar ou requerer o uso do módulo criptográfico para garantir a proteção dos dados.

- *Data Encryption Standard (DES)*: este algoritmo utiliza criptografia de chaves simétricas para proteger as comunicações através da rede.

• *Triple DES (3DES)*: esta variante, criptografa a mensagem com três passos do algoritmo DES. Ele provê um alto grau de segurança na mensagem, porém tem como ponto negativo a perda de performance devido ao processamento adicional. Em geral é três vezes mais lento que o DES.

Para verificar a integridade dos dados, o banco *Oracle* utiliza os seguintes algoritmos:

• *MD5 Checksum*: este algoritmo provê a integridade dos dados através do *hash* e da seqüência para garantir que o dado não foi alterado ou roubado quando é transmitido pela rede.

• *Secure Hash Algorithm (SHA):* é similar ao MD5, porém é indicado para grandes mensagens que necessitam de alta segurança.

Na Figura 6.6 é demonstrado como se pode configurar uma BDE para acesso nativo ao banco de dados utilizando os parâmetros definidos no servidor para comunicação.

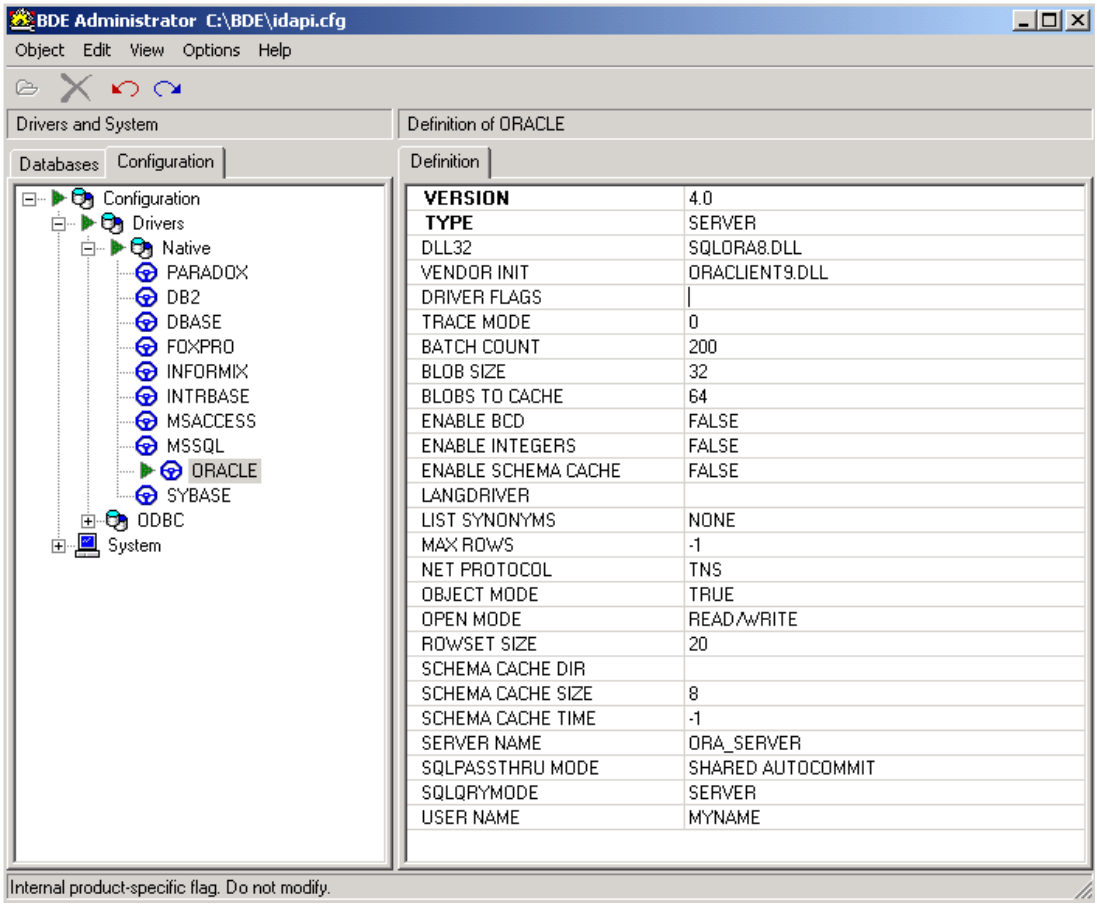

Figura 6.6 – Definindo parâmetros na BDE para Banco de Dados Oracle 9i

Uma vez que todos os elementos envolvidos na comunicação estão definidos e conhecidos, bem como os métodos que se pode utilizar para garantir a integridade dos dados e sua origem, tem-se na Figura 6.7 todo o cenário da utilização da aplicação Cliente/Servidor através de uma rede pública.

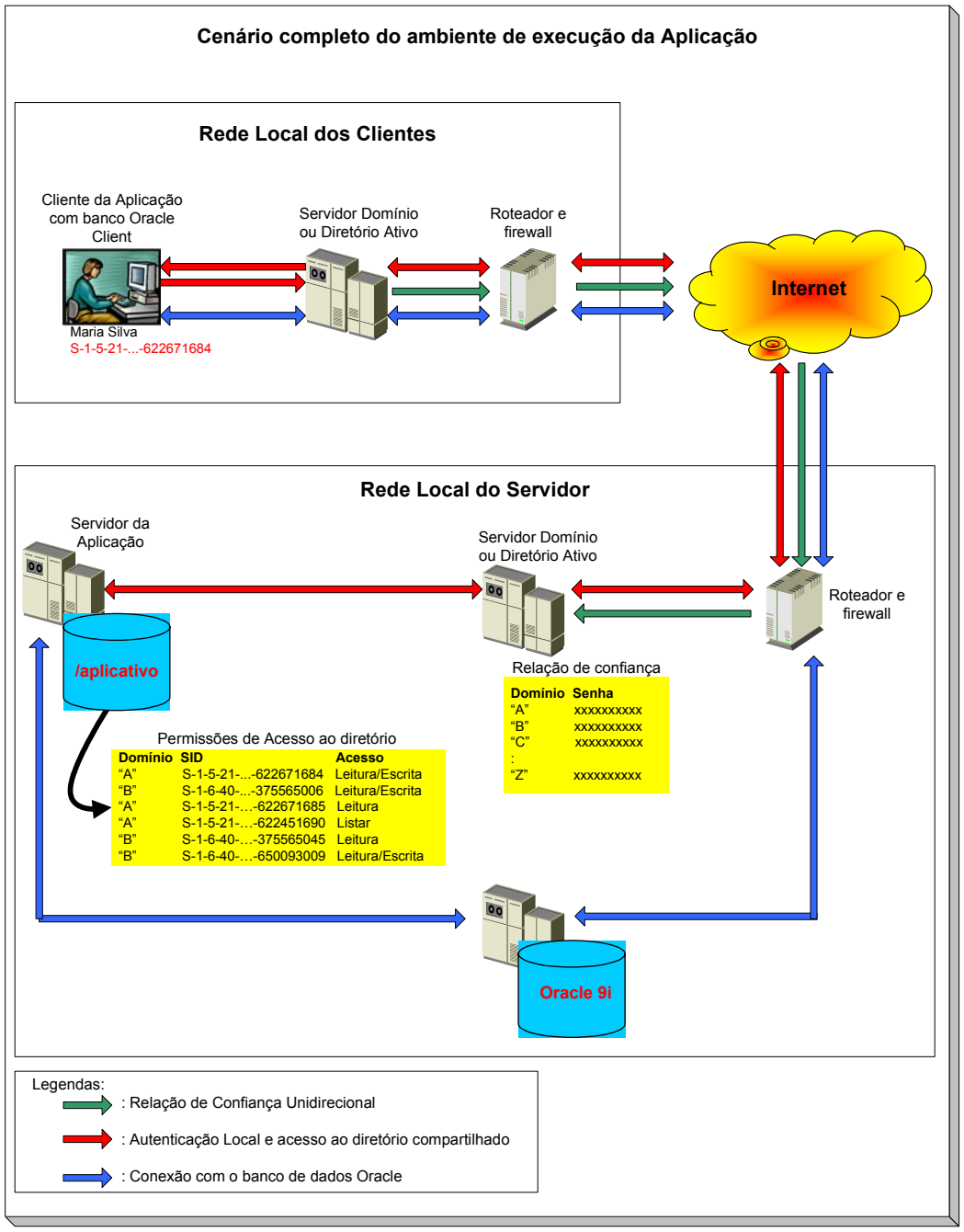

Figura 6.7 – Cenário da solução com Relacionamento de Confiança e *IPSec*

As portas TCP e UDP utilizadas no cenário apresentado na Figura 6.7 estão listadas na Tabela 6.2 a seguir.

Porta	Transporte	Finalidade
	ESP[1]	Provê a confidencialidade para o *payload* IP. ESP no modo transporte não criptografa todo o pacote IP, e sim apenas o *payload*
88	UDP	Kerberos
135	TCP	PortMapper – Porta fixa do Netlogon
138-139	TCP	NetBEUI – Mapear diretórios remotos
389	TCP e UDP	LDAP – Utilizado pelos servidores com *Windows 2000/2003 Server*
445	TCP	SMB – Estabelecimento do Relacionamento de Confiança
500	UDP	Utilizado nas comunicações em *IPSec*

Tabela 6.2 – Portas utilizadas na comunicação cliente/servidor do cenário da figura 6.7

Através do conhecimento das portas de comunicação utilizadas pelos serviços, tanto na origem quanto no destino, podemos definir regras nos *firewalls* que permitam a passagem de pacotes apenas de redes conhecidas, adicionando-se mais um fator para garantir a segurança.

Neste cenário a administração do acesso é dividida em duas partes, os clientes são administrados pelos administradores locais, eles definem quais são os usuários que farão parte dos grupos e, portanto podem definir quem pode utilizar ou não os recursos no servidor. A segunda parte fica a cargo do administrador do servidor de arquivos que precisa confiar na informação que busca no domínio/diretório ativo remoto.

6.4 Análise da solução com Certificação Digital

Outra opção avaliada é com a utilização de certificados digitais. Estes certificados podem ser gerados e administrados através de entidades certificadoras dentro da própria organização, criando uma hierarquia de certificados e um ambiente computacional mais seguro.

No caso de aplicações cliente/servidor, os certificados não são utilizados para garantir o acesso aos recursos compartilhados, por tratar-se da plataforma *MS-Windows.* Apenas o *SID* é usado para esta finalidade, entretanto, seu uso é bastante útil na versão para WEB da aplicação, pois há possibilidade de

[1] *Encapsulating Security Payload*

certificar tanto o usuário do sistema, quanto a sua estação de trabalho, provendo autenticidade e criptografia no canal de comunicação entre o cliente e o servidor da aplicação.

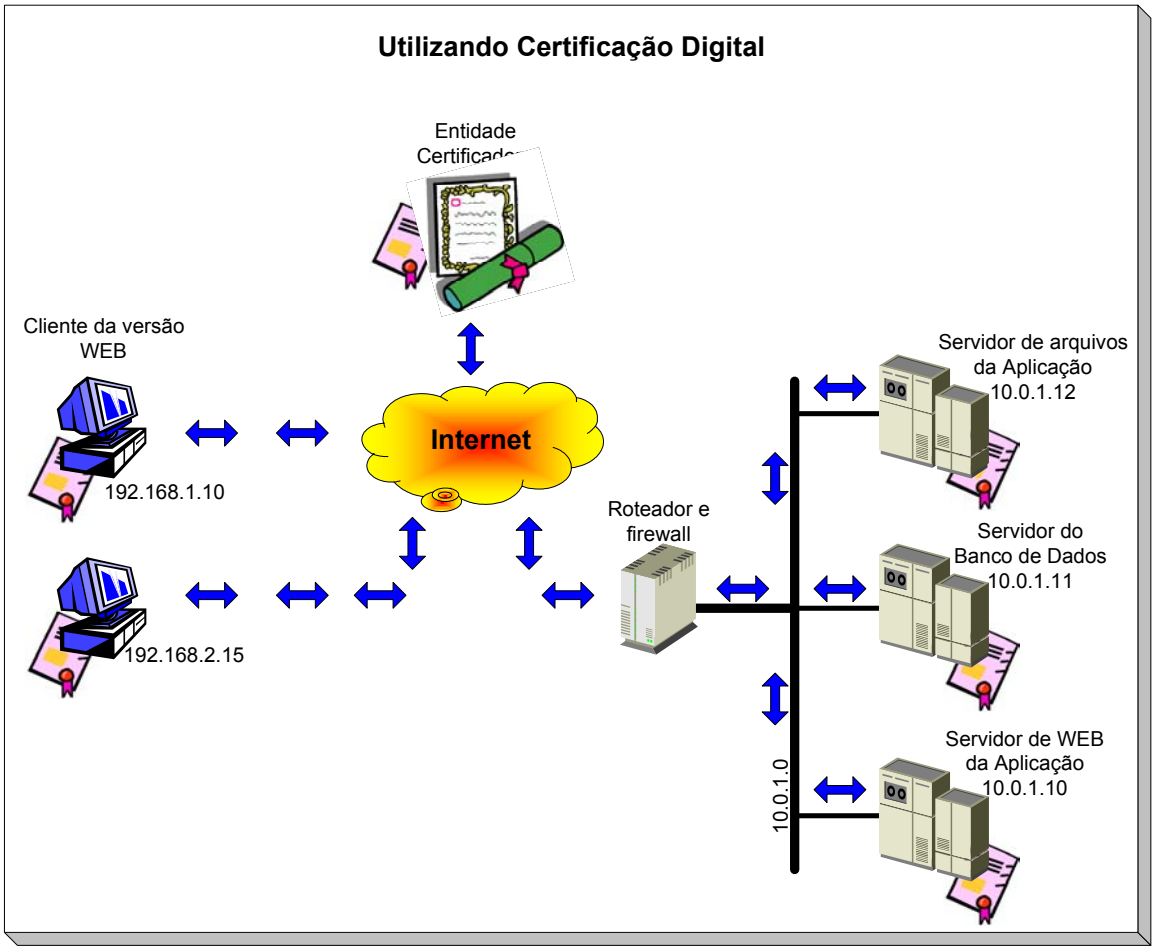

Figura 6.8 – Cenário da solução com Certificação Digital

A implementação da estrutura é bastante facilitada pelo serviço de Certificação Digital. Existem duas opções para se adquirir um certificado: via uma página *web* ou utilizando a estrutura do diretório ativo.

O administrador mantém uma lista dos certificados válidos e também dos certificados revogados. Esta lista deve ser publicada para que todos envolvidos na solução tenham conhecimento.

O algoritmo RSA é utilizado para criar a assinatura digital [42] e em seguida é anexado aos dados originais da mensagem. Este procedimento é ilustrado na Figura 6.9.

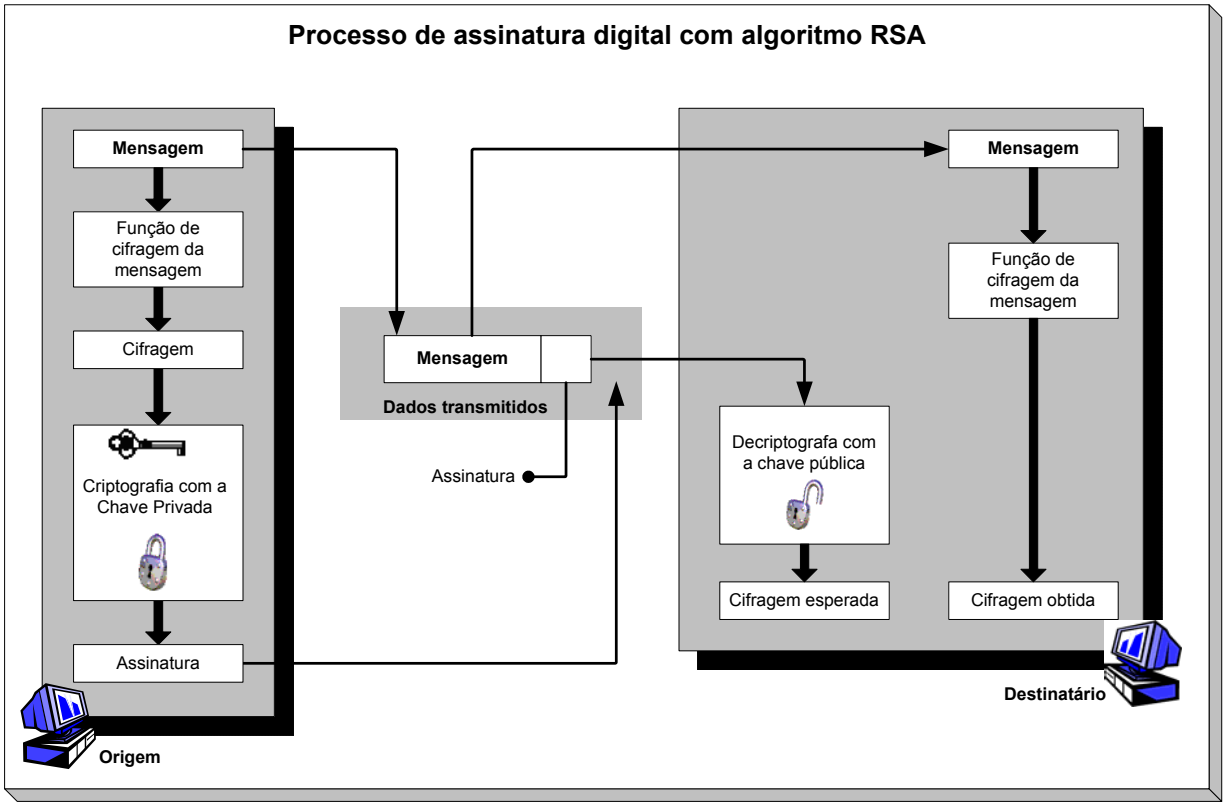

Figura 6.9 – Utilização do algoritmo RSA para assinatura digital

Uma vez realizado o processo para assinatura digital a informação pode ser transmitida pela rede através de um canal seguro, utilizando o protocolo *IPSec*, por exemplo. Na Figura 6.10 baseada em [43] pode-se perceber os elementos envolvidos para utilização do serviço de Certificados Digitais e a infra-estrutura de chaves públicas aplicadas por servidores *MS-Windows 2000/2003 Server*.

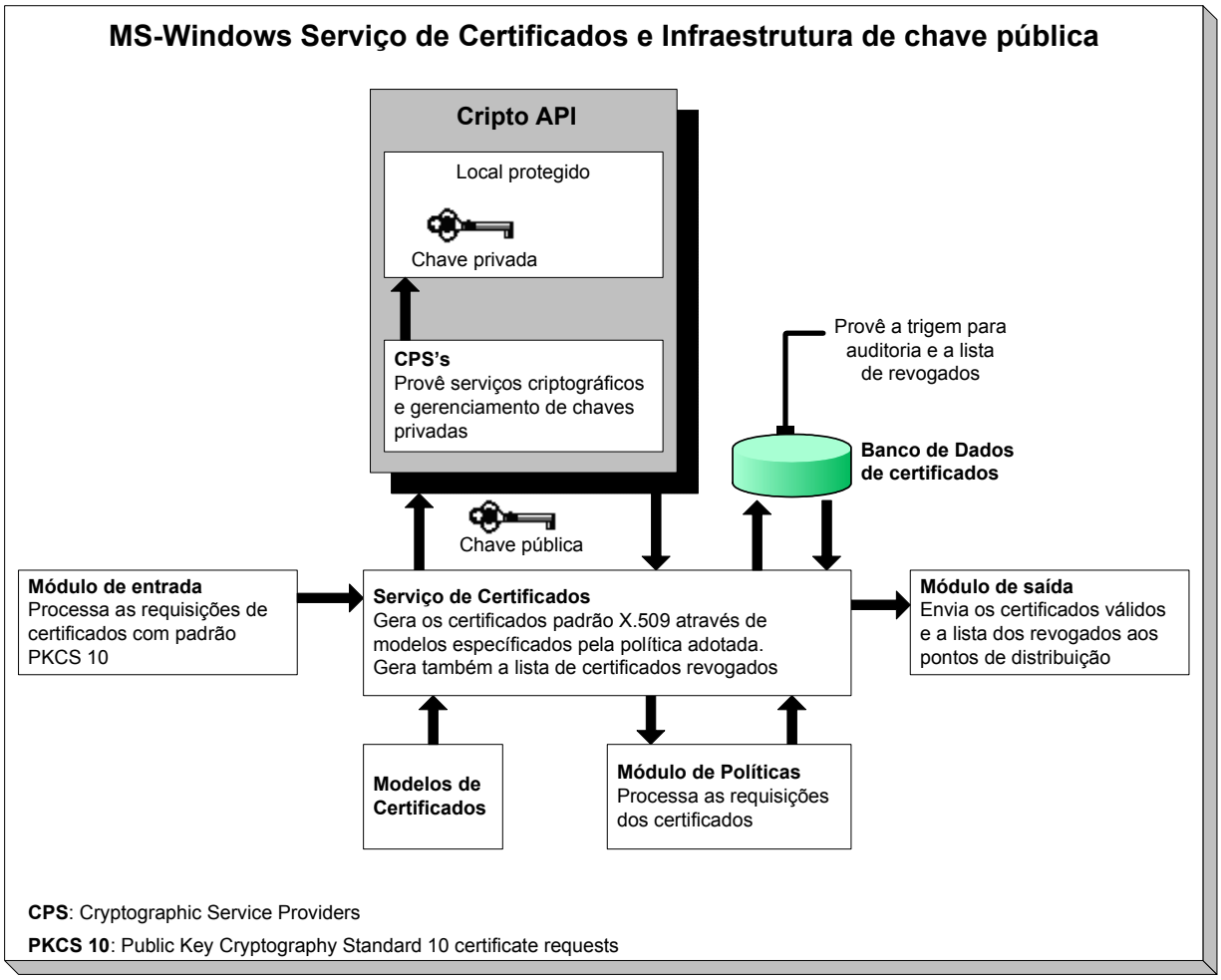

Figura 6.10 – Implementação do Serviço de Certificados Digitais na plataforma *MS-Windows*

6.5 Proposta de solução com *Proxy* Reverso

De maneira geral as aplicações desenvolvidas para execução em ambiente WEB tendem, por uma questão sobrevivência, a serem compatíveis com pelo menos dois dos principais *WEBServers* existentes e bem conhecidos no mercado. O APACHE, de domínio público e código aberto e o MS-IIS (*Microsoft Internet Information Service*) desenvolvido pela *Microsoft* e disponível a partir das versões *Server do Windows NT*.

Evidentemente existem casos, como aplicações desenvolvidas na linguagem Java, por exemplo, que necessitam além do APACHE também do *software* TOMCAT e um *runtime* Java instalados no servidor para que funcionem.

A implementação de segurança para aplicações disponível via Internet ocorre através do protocolo *https* ao invés do tradicional *http*, ou seja, utiliza-se o protocolo SSL (vide item 4.1.3). Desta maneira agrega-se criptografia na comunicação, gerando um canal seguro entre o cliente e o servidor da aplicação, todavia, como é o caso da aplicação em estudo, é freqüente o uso do servidor de páginas MS-IIS, que conhecidamente é alvo de diversos tipos de ataque que podem causar a indisponibilidade do serviço de WEB ou a troca de páginas da empresa, por exemplo.

Apesar do grande empenho da *Microsoft* para eliminar as falhas, detectadas desde o desenvolvimento do produto e a implementação do protocolo SSL nas versões mais recentes, muitos administradores procuram soluções que evitam o acesso direto as páginas neste servidor, pois tratam-se de dados críticos e muitas vezes sigilosos, neste caso existe uma outra maneira de prover uma camada de segurança e evitar o acesso direto a estes servidores *web*. Esta técnica é denominada *proxy* reverso.

O *proxy* reverso funciona como um receptor de páginas para um ou mais domínios Internet registrados no DNS, ou seja, ele pode receber todas as solicitações enviadas ao servidor de páginas original como um *front-end* do *site*, e somente através dele estas solicitações chegam ao servidor real da aplicação [03].

Pode-se somar a esta proposta as vantagens obtidas através do uso dos certificados digitais. No entanto a estrutura computacional (com a adição de um serviço de certificação interno ou externo) e a administração do ambiente, tendem a se tornarem mais complexos. A Figura 6.11 demonstra o esquema da comunicação entre o cliente e o servidor da aplicação *web*, bem como a ação do *proxy* reverso.

O *proxy* reverso pode atuar como um filtro de acesso ao servidor da aplicação. Em geral é utilizado o software APACHE, devido a facilidade de instalação e configuração dos arquivos para atuar como um *proxy* reverso [03].

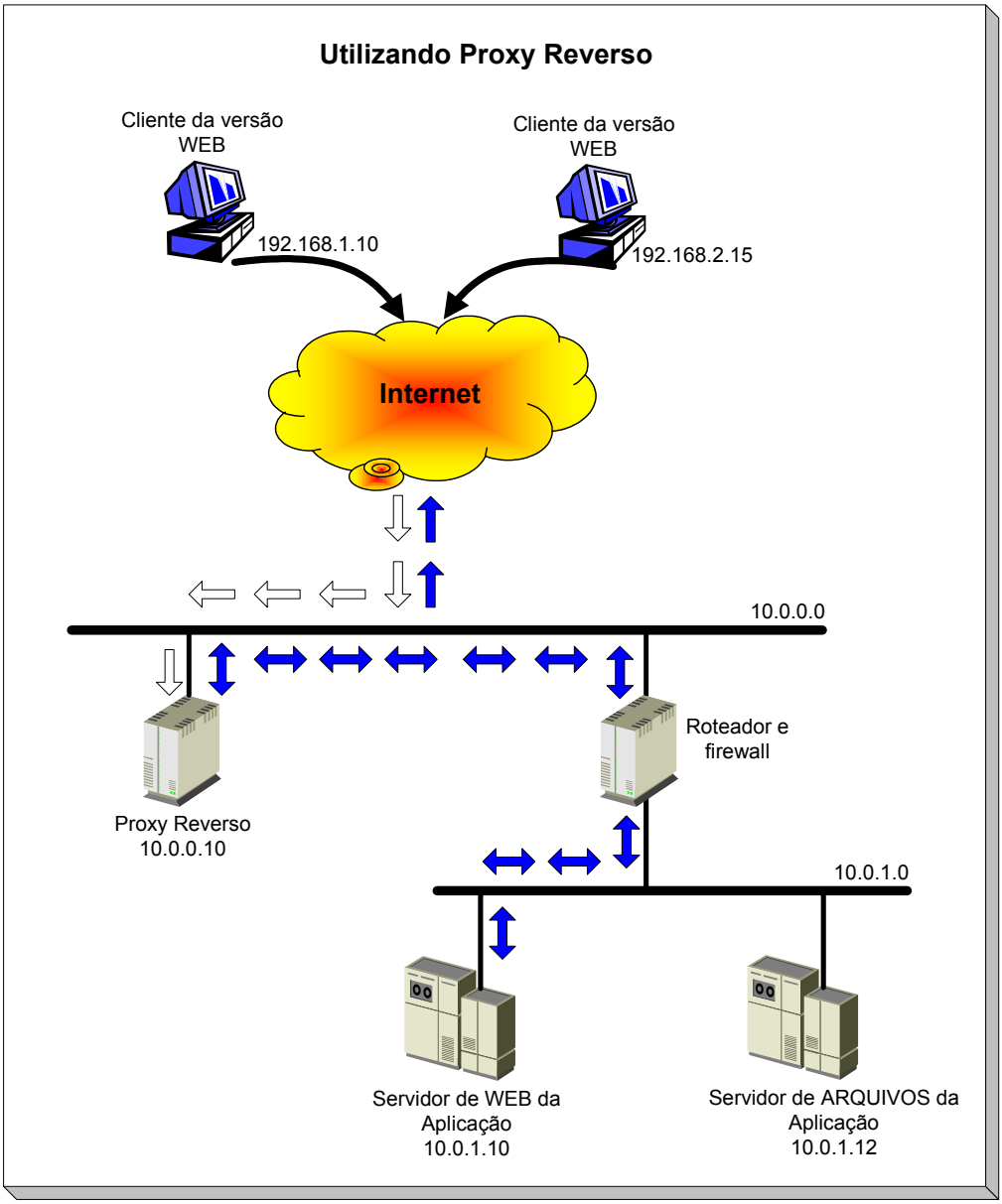

Figura 6.11 – Utilizando *Proxy* Reverso para proteger o servidor WEB interno

6.6 Comparação entre as soluções

Uma vez conhecidas as propostas para solução do problema, é importante analisar com mais detalhes os aspectos positivos e negativos de cada proposta. A Tabela 6.3 apresenta um resumo com estas informações.

Proposta	Aspectos Positivos	Aspectos Negativos
VPN *Microsoft*	• Integrada ao Sistema Operacional • Gerenciamento do acesso aos recursos do servidor e dos usuários centralizado • Provê autenticidade e segurança entre o cliente e o servidor • Pode ser substituída por VPN de software livre • Cliente pode possuir qualquer Sistema Operacional *Windows* de 32 bits	• Administração Complexa • Degradação da performance com crescimento de usuários (escalabilidade) • Implementação complexa no cliente e no servidor • Protocolos adicionais são utilizados para realização da VPN Problemas para configuração do Banco Oracle com endereços de VPN
VPN com *IPSec*	• Implementação simples com protocolo nativo • Gerenciamento dos usuários é descentralizado nos domínios • Gerenciamento do acesso ao recurso centralizado • Provê autenticidade e segurança entre o cliente e o servidor • Não necessita da criação da VPN de camada de enlace; a autenticação na rede local é utilizada para prover autenticidade	• Confiança em domínios/diretórios ativos remotos • Clientes devem possuir Sistema Operacional que implemente *IPSec* • Não possui autonomia para aplicar políticas de segurança de senha nos domínios confiáveis
Certificação Digital	• Simples implementação do serviço • Implementa conexão segura através da WEB • Garante através da infra-estrutura de chave pública a identidade dos participantes na comunicação • Pode ser utilizado em conjunto com o serviço de diretório ativo • Maior controle e conhecimento dos computadores que acessam o sistema	• Desnecessário para acesso aos recursos compartilhados em ambiente Windows • Necessita criar uma hierarquia de entidades certificadoras • Necessita do IIS para implementar de forma integrada ao Sistema Operacional

Proxy Reverso	• Implementa conexão segura através da WEB • Não permite acesso direto ao servidor WEB da aplicação • Pode ser implantada com software livre	• Necessita de mais um servidor WEB • Implementação requer conhecimento na técnica de *proxy* reverso em conjunto com *firewalls*

Tabela 6.3 – Quadro comparativo entre as propostas de solução

6.7 Análise da solução adotada para estudo de caso

Através da comparação obtida nos estudos e testes realizados em cada uma das propostas, pôde-se adotar uma solução adequada para resolver os problemas de comunicação segurança, autenticidade, compatibilidade e escalabilidade para aplicações do tipo cliente/servidor com características semelhantes de nosso estudo de caso.

Deve-se ressaltar contudo, que todas as propostas são viáveis para resolver o problema. A adoção de uma em particular não exclui sua aplicação em outros casos, dados como número de usuários que acessam a aplicação, quantidade e qualidade do tráfego gerado na rede e ambiente computacional são fundamentais para a escolha.

Em geral, aplicações cliente/servidor são executadas através de redes geograficamente distantes, e não é difícil encontrar empresas/instituições que utilizam as redes públicas ou inseguras para a troca de informações, sendo assim, a solução a ser adotada necessita, além de fornecer os serviços fundamentais para garantir a autenticidade e a segurança da informação entre os integrantes da comunicação, também, e principalmente, mecanismos que possam prover a facilidade de implementação, tendo o cliente como foco principal e, a administração do sistema como um todo.

Outra questão de grande importância é a escalabilidade, a tendência é que estes sistemas cresçam através de suas manutenções e aprimoramentos, bem como a quantidade de usuários que utilizam a aplicação para suas rotinas diárias, por conseqüência o aumento gradativo dos computadores que necessitam acessar os recursos compartilhados pelo servidor. Somando-se a isso deve-se pensar na questão da evolução tecnológica e dos novos Sistemas Operacionais, permitindo que as implementações possam ser compatíveis com estas novas tecnologias.

6.7.1 Fatores para adoção da proposta de VPN com *IPSec* nativo

Dentre as propostas analisadas, a montagem de uma VPN entre o cliente e o servidor da aplicação apresentou-se mais vantajosa para a versão cliente/servidor, desde que não seja utilizada nenhuma solução com o modelo de VPN implementada como um serviço do Sistema Operacional.

Isso se deve ao fato das características do modelo de rede local implementado pelo *Microsoft*, que utiliza as funções do protocolo NetBIOS/NetBEUI, encapsuladas em pacotes TCP/IP para implementação de recursos compartilhados e o método de identificação do usuário (SID) como um passaporte para acesso a estes recursos. Com estas restrições, torna-se inevitável o uso da técnica de Relação de Confiança entre domínios ou Diretórios Ativo para integrar usuários de redes fisicamente distintas. Sem este recurso, os usuários que necessitam mapear uma unidade de disco remoto precisam ser autenticados novamente a cada nova sessão, como pode ser visto na Figura 6.12 ou, mapear o recurso através de comandos na janela de *prompt* do MS-DOS, como exemplificado na Figura 6.13, antes de iniciar o uso da aplicação.

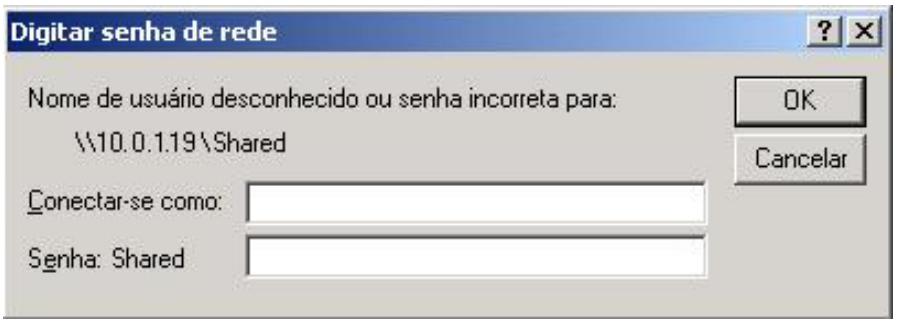

Figura 6.12 – Janela de autenticação do usuário para acesso a recurso remoto

Figura 6.13 – Comando no *prompt* do MS-DOS para mapear um disco remoto

Estes procedimentos são um incômodo para quem utiliza a aplicação. Com o uso do relacionamento de confiança, o administrador do recurso compartilhado pode escolher os

usuários/grupos, bem como os privilégios de acesso personalizados, eliminando a necessidade de novas autenticações. Portanto o uso do recurso passa a ser transparente para o usuário.

O uso do protocolo *IPSec* nativo implementado nos Sistemas Operacionais *MS-Windows* 2000 Profissional em diante, torna segura a comunicação entre o cliente e os servidores com os quais interage, isto também proporciona que Sistemas Operacionais obsoletos, como *MS-Windows 95/98/98SE* e *Millenium*, sejam substituídos para atender a este requisito, eliminado desta maneira a possibilidade de ataques a estas versões que não possuem suporte para os protocolos seguros. A simplicidade e a facilidade para implementação da solução é outra vantagem com relação as demais soluções, relacionamentos de confiança são simples de se fazer até mesmo utilizando software de domínio público, como o SAMBA[1].

O fato de ser unidirecional não interfere na política de administração dos usuários locais, pois os usuários são apenas exportados, não tendo nenhuma interferência externa ou necessidade de modificação em grupos, privilégios e outros aspectos no domínio confiável. Este processo é realizado apenas uma vez entre os domínios confiantes e confiáveis, pois uma vez estabelecida a relação de confiança, os sistemas operacionais possuem mecanismos para verificar os estados desta relação e alertar em caso de falhas, desta maneira a administração torna-se descentralizada, pois os administradores locais conhecem melhor sua realidade que o administrador do ambiente servidor. Isso proporciona a escalabilidade necessária para adicionar novas redes para acesso a aplicação, independente de sua localização geográfica, de maneira organizada e mensurável.

Para a solução da versão WEB a implementação de um *proxy* reverso mostrou-se mais eficaz. Apesar de adicionar um novo elemento no ambiente computacional, o que também ocorre com a solução utilizando certificados digitais, a performance não foi prejudicada.

Mais uma vez a simplicidade da solução e a transparência para os usuários foram os fatores determinantes para a escolha, além, é claro, da questão da segurança com o uso do protocolo SSL e também a proteção ao acesso direto ao servidor primário das páginas da aplicação, evitando a exposição para ataques realizados por *hackers* através das vulnerabilidades conhecidas nesse tipo de serviço.

Em geral versões para WEB de aplicações desenvolvidas para fins administrativos, tendem a serem amplamente utilizadas para consultas, com pouca entrada de dados. Este é o perfil em que nosso estudo de caso se encaixa, não necessitando obrigatoriamente de um esquema complexo de certificados digitais.

[1] Somente a partir da versão 3.0.1 do SAMBA *Server* é possível fazer relacionamentos de confiança com domínios *MS-Windows*

6.7.2 Fatores para eliminação da proposta com VPN *Microsoft*

Esta solução possui grandes vantagens com relação a segurança e autenticidade. A criação de um túnel criptográfico através de uma rede insegura torna praticamente inviável a decodificação da informação e conseqüentemente o acesso ao texto original a membros não participantes da comunicação.

É importante esclarecer que a solução exige um conhecimento avançado no modelo de VPN como serviço do Sistema Operacional. Métodos para cadastramento de usuários e distribuição dos endereços entre os componentes da VPN devem ser estabelecidos. Não se pode esquecer que dentro desta VPN, além dos clientes, devem constar todos os servidores utilizados pela aplicação, isso inclui também os servidores nas redes locais destes clientes, que são utilizados para estabelecer a confiança entre os domínios.

Por se tratar de uma estrutura complexa, deve-se criar ações e mecanismos para a gestão dos endereços de rede --- em geral uma classe de endereços reservados --- da VPN distribuídos pelo concentrador, estes devem ser pré-determinados pelo administrador da VPN aos servidores com o qual são estabelecidas as Relações de Confiança, para que elas não se quebrem ao desconectar um servidor por qualquer motivo, bem como garantir que os servidores da aplicação e banco de dados possuam sempre o mesmo endereço. Além disso, os administradores precisam gerar rotinas ou mecanismos que possibilitem a entrada automática dos servidores na VPN e terem uma preocupação especial no caso de manutenções, a fim de que não haja perda de informações desta Relação de Confiança, evitando que todo o esquema de comunicação com o servidor central necessite ser refeito.

A questão da administração do ambiente torna-se complexa, tanto para os clientes, quanto para os administradores, pois deve-se levar em consideração que a medida em que novos clientes são adicionados a VPN, bem como os novos servidores, adiciona-se uma nova carga de processamento nos elementos envolvidos na solução. As contas dos clientes são centralizadas no concentrador de VPN e todo o processo de autenticação e definição dos endereços ocorre neste equipamento.

Em virtude destes fatos este *hardware* deve possuir características físicas capazes de atender a demanda crescente sem prejudicar a performance da aplicação, outro fator administrativo de dificulta a implementação está no uso do banco de dados, em geral estes bancos necessitam de uma *string* de conexão com informações sobre o nome na rede do servidor do banco de dados e as características da instância da base instalada (nome, porta de conexão e outros parâmetros para conexão a instância do banco).

Uma vez criada esta instância no servidor os clientes utilizarão estes parâmetros para acesso, sendo assim os administradores da base deverão fazer suas manutenções integrados a esta rede ou, possuírem acesso direto ao console do servidor.

Também existem problemas para compatibilidade do serviço de VPN com sistemas operacionais não-*Microsoft* ou mesmo proprietário, em muitos casos é difícil, ou mesmo inviável, a conexão do servidor a VPN criada, pode-se citar como exemplo um servidor com banco de dados *Oracle9i*, instalado sobre uma plataforma *HP-Tru64 Unix* em uma máquina *AlphaServer*[1]. Portanto, a implementação desta solução é menos laboriosa quando se utiliza apenas recurso das redes e dos Sistemas Operacionais da *Microsoft*, inclui-se nesta proposta o servidor do Banco de Dados. Como compensação pode-se utilizar a aplicação em qualquer um dos Sistemas Operacionais da *Microsoft* de 32bits, pois estes já possuem suporte para conexão a uma VPN.

Não se pode esquecer que por se tratar de uma solução com algoritmos criptográficos, exige dos clientes e servidores um processamento adicional para a conexão a VPN e também para a geração dos pacotes que são transmitidos pelo túnel, em *hardwares* mais antigos isso pode ser representativo na performance da aplicação.

6.7.3 Fatores para eliminação da proposta com Certificação Digital

A solução através do uso dos Certificados Digitais provê um grau de segurança e autenticidade irrevogável, com a utilização da infra-estrutura de chaves públicas pode-se certificar todos os elementos envolvidos na comunicação, principalmente os clientes e os servidores da aplicação. Os Certificados Digitais são um mecanismo poderoso, contudo a implementação deste serviço requer do administrador um conhecimento das técnicas criptográficas utilizadas e, a melhor maneira para seu aproveitamento na solução.

A utilização deste serviço mostrou-se eficiente como solução para a versão WEB da aplicação, já para a versão cliente/servidor não pôde ser aplicada, com exceção dos ambientes em que o serviço de Diretório Ativo pôde ser instalado, mesmo assim como um item de segurança adicional. Isto ocorre devido ao fato que em redes *Microsoft*, os recursos compartilhados são liberados após a checagem do SID do usuário e esta informação não é transmitida juntamente com o Certificado.

Desta maneira para implementação desta solução é necessário, além da criação da estrutura de entidades certificadoras, também todo o esquema envolvendo o Relacionamento de Confiança entre os

[1] Plataforma da Antiga *Digital Corporation* adquirida pela *Hewlett Packard* (HP), com processadores RISC de 64bits

domínios do cliente e do servidor da aplicação. Isso torna a solução administrativamente mais complexa, pois requer a adoção de uma política para a solicitação e aprovação dos certificados, sejam elas para usuários ou computadores e, também a distribuição da lista dos certificados revogados, pois essa informação precisa ser utilizada pelo administrador para realizar o compartilhamento e a liberação do acesso para o cliente, neste sentido é recomendável a criação de rotinas para automatizar este procedimento.

No caso da adoção da solução, é de fundamental importância a criação de uma entidade certificadora reserva, com a cópia do banco de dados de certificados da entidade principal.

Conclusão

Nos últimos anos a preocupação com a segurança da informação tornou-se fator fundamental para a implantação de sistemas nas empresas. A exposição das vulnerabilidades dos protocolos utilizados e o conhecimento de seus riscos fizeram com que ferramentas para agregar segurança, não somente ao meio de transmissão, mas também nos meios de armazenamento, fossem adotadas e amplamente analisadas.

Atualmente o que se vê é um grande mercado oferecendo aplicações, com as mesmas características encontradas em nosso estudo de caso, sendo comercializadas, sem que a solução completa seja apresentada para sua implantação, muitas vezes por falta deste conhecimento dos próprios desenvolvedores.

Mecanismos simples de serem adotados na área de segurança podem trazer grandes benefícios. O uso do protocolo *IPSec* é uma prova disso. No cenário de rede atual, existem milhares de programas na Internet que facilitam o trabalho dos atacantes, simulam, decodificam e alteram dados e se valem justamente das vulnerabilidades conhecidas e informações capturadas para realizar um ataque.

O conhecimento das principais características dos Sistemas Operacionais e seus serviços disponíveis, focados especialmente na segurança e autenticidade, podem fornecer informações importantes para a decisão de colocar uma aplicação que utiliza a rede para troca de informações.

Capítulo 7

Conclusão

O desenvolvimento de aplicações sofreu grandes modificações ao longo das últimas décadas. O cenário de execução transformou-se rapidamente, deixando os antigos ambientes homogêneos -em sua maioria composta por *mainframes* e terminais com tecnologia proprietária das empresas, passando para redes não-proprietárias e protocolos abertos.

Pode-se destacar também outros fatores como o surgimento e a adoção da Internet pelas empresas e instituições e do estabelecimento do protocolo TCP/IP como padrão de *facto*. Além disso, a significativa evolução do *hardware* que proporcionou a diminuição dos custos com infra-estrutura de comunicação e o aumento da velocidade no processamento das informações através da comutação de pacotes em taxas, antes inimagináveis.

Acompanhando a evolução deste cenário, foram desenvolvidos no modelo cliente/servidor novas arquiteturas e modelos capazes de atender as necessidades das novas aplicações, contudo não se pode esquecer que os avanços na área de segurança permitiram a integração entre cliente e servidor atendendo os princípios básicos que regem a comunicação descrita no item 3.2.4.

A apresentação do estudo de caso demonstrou, na prática, os problemas que são encontrados para a implementação e execução das aplicações no cenário da comunicação, bem como as opções disponíveis que podem ser aplicadas e adaptadas a fim de solucionar um problema do mundo real.

Obteve-se com este trabalho importantes contribuições para se compreender melhor o modelo Cliente/Servidor e sua arquitetura, vista em detalhes no Capítulo 2, tornando-se uma referência aos que se interessam e necessitam de mais informações sobre o assunto.

No Capítulo 3 desenvolveu-se o cenário da comunicação, suas nomenclaturas e tendências utilizadas para a construção de aplicações voltadas ao mercado e grandes corporações. Também foram expostas informações com as características dos protocolos de comunicação e suas particularidades, destacando-se os protocolos TCP/IP, SMB e NetBIOS/NetBEUI. Além disso, mostrou-se os aspectos importantes da conexão física/lógica entre sistemas e informações sobre sincronismo e passagem de mensagens utilizadas para comunicação.

Os Capítulos 4 e 5 foram complementares, contendo informações fundamentais e detalhadas sobre os protocolos seguros, utilizados para interligação dos clientes aos servidores. No capítulo 4 destacou-se a exposição de protocolos seguros como SSL e *IPSec* e mecanismos para fornecer autenticação como KERBEROS. O estudo destes proporcionou sua adoção na proposta de solução de segurança no estudo de caso, apresentado no Capítulo 5.

No Capítulo 5 foi apresentado um estudo de caso utilizando uma aplicação Cliente/Servidor comercial voltada para área administrativa. Desenvolvida com ferramentas e linguagens para execução na plataforma *MS-Windows*, que é sem dúvida a plataforma mais comum nas empresas/instituições. Analisando a aplicação e seus requisitos, pode-se constatar que a implementação segura destes produtos, na plataforma para qual foi desenvolvida, requer um estudo mais elaborado, pois o cenário apresentado é repleto de falhas e vulnerabilidades que podem ser exploradas.

Sendo assim, as tecnologias apresentadas no Capítulo 5 utilizando técnicas de tunelamento com a criação de VPN's, Relacionamentos de Confiança e Certificação Digital dos clientes e servidores, em conjunto com os protocolos seguros expostos no Capítulo 4, forneceram subsídios para elaboração das propostas de solução que foram apresentadas no Capítulo 6.

Neste sentido o presente trabalho contribuí aos administradores que necessitam implementar aplicações com características semelhantes às encontradas em nosso estudo de caso, de forma segura e que possa garantir a autenticidade dos usuários, mesmo utilizando conexões através de redes inseguras.

Por fim no capítulo 6 foi desenvolvida uma proposta viável e segura para a implementação da aplicação de nosso estudo de caso. Utilizou-se conhecimentos e técnicas vistas ao longo deste trabalho e apresentou-se também as razões pelas quais soluções válidas não puderam ou não foram recomendadas. Por se tratar de uma aplicação real concebida com um modelo amplamente difundido, pode-se aplicar esta proposta de solução em diversas aplicações que possuam o mesmo perfil e funcionalidade.

Este trabalho possibilitou uma pesquisa abrangente da arquitetura cliente/servidor e apresentou propostas para adição de camadas de segurança, tendo como cenário, redes distintas utilizando conexões remotas ou redes inseguras. Como resultado das pesquisas pôde-se demonstrar que em muitos casos Sistemas Operacionais e serviços de rede - muitas vezes proprietários, podem ser substituídos, sem perda alguma, por soluções não-proprietárias como, por exemplo, UNIX.

Serviços como WEB, VPN, Arquivos e até servidores de domínios, mesmo para redes *Windows*, já existem para este ambiente. Deste modo abre-se um novo horizonte para adequação do ambiente corporativo. Com o grande desenvolvimento das aplicações *freeware* para UNIX, não será obrigatório o uso de apenas um modelo único para a execução de aplicações Cliente/Servidor.

7.1 Trabalhos futuros

Ao longo deste trabalho foram expostos os elementos principais da arquitetura e do modelo cliente/servidor. Como resultado das pesquisas realizadas para resolver um problema comum a plataforma onde é executada a Aplicação, descrita em nosso estudo de caso do Capítulo 5, foram apresentadas uma série de propostas viáveis, considerando-se todo o ambiente, principalmente o meio de comunicação. Elemento este, conhecido e composto por protocolos de rede amplamente utilizados, porém com restrições na área de segurança.

Incluiu-se nestas pesquisas aspectos que abrangem desde o Sistema Operacional utilizado tanto pelos clientes, quanto pelos servidores das aplicações, até a tecnologia de rede empregada para interconexão, sejam elas institucionais, particulares ou públicas.

Dentre as soluções pesquisadas a utilização de uma VPN como serviço do Sistema Operacional, deveria ser considerada uma solução natural para o problema. Contudo, não se mostrou eficaz devido a complexidade de sua implementação e administração, também por uma particularidade como o processo de autenticação é realizado em redes *Microsoft*.

Sendo assim existe um campo a ser explorado, de forma a proporcionar que as autenticações realizadas através do serviço de VPN instalado em um servidor --- existente tanto em plataformas proprietárias, quanto em *software* livre --- possam ser utilizadas para acesso aos recursos compartilhados do Sistema na parte Servidor da aplicação. Desta forma pode-se viabilizar o uso de aplicações construídas para execução com tecnologia Cliente/Servidor através de túneis seguros, sem a necessidade da criação de mecanismos adicionais, estruturas complexas ou *upgrade* de *software* ou

hardware em máquinas clientes, diminuindo e simplificando significativamente os custos de implantação de aplicações cliente/servidor através de uma rede de computadores pública/privada.

Glossário de siglas

ADSP: Appletalk Data Stream Protocol

AH: Authentication Header

API: Application Programming Interface

APPC: Advanced Program-to-Program Communications (IBM)

ATP: Appletalk Transaction Protocol

BDE: Borland Database Engine

BI: Business Intelligence

CA: Certificate Authority

CRL: Certification Revocation List

CSS: Cascading Style Sheets

DDP: Datagram Delivery Protocol

DES: Data Encryption Standard

DES-CBC: Data Encryption Standard-Cipher Block Chaining

DSS: Decision Support System

DSS: Digital Signature Standard

EIS: Executive Information System

ERP: Enterprise Resource Planning

ESP: Encapsulated Security Payload

FS DRIVERS: File System Drivers

GRE: Generic Routing Encapsulation

GUI: Graphical User Interface

HMAC: Keyed-Hashing Message Authentication Code

IDEA: International Data Encryption Algorithm

IDP: Internet Datagram Protocol

IEEE: Institute of Electrical Engineers

IETF: Internet Engineering Task Force

IIS: Internet Information Service (Microsoft Web Server)

IKE: Internet Key Exchange

IP: Internet Protocol

IPC: InterProcess Comunication

IPX: Internet Protocol Exchange

ISAKMP: Internet Security Association and Key Management Protocol

ISDN: Integrated Services Digital Network

ISP: Internet Service Provider

LDAP: Lightweight Directory Access Protocol

LLC: Logical Link Control

LPC: Local Procedure Call

LSASS: Local Security Authority Subsystem Service

LU: Logical Unit

LU: Logical Unit (IBM)

MAC: Message Authentication Code

MD5: Message Digest Five

MSGINA: Microsoft Graphical Identification and Authentication

NBDD: NetBIOS Datagram Distribution

NBNS: NetBIOS Name Server

NCSA: National Center for Supercomputing Applications

NTLM: NTLanMan (Windows NT Lan Manager Authentication Protocol)

OLAP: OnLine Analytical Processing

OS2: Operation System Two (IBM)

OSI: Open System Interconnection

PC: Personal Computer

PPP: Point to Point Protocol

PPTP: Point to Point Tunneling Protocol

PU: Physical Unit (IBM)

RADIUS: Remote Authentication Dial-In User Service

RAS: Remote Access Service

RFC: Request for Comments

RPC: Remote Procedure Call

SDLC: Synchronous Data Link Control

SGDB: Sistema Gerenciador de Banco de Dados

SHA: Secure Hash Algorithm

SID: Security Identifier

SMB: Server Message Block Protocol

SNA: System Network Architecture

SO: Sistema Operacional

SOR: Sistema Operacional de Rede

SPI: Security Parameter Index

SPP: Sequenced Packet Protocol

SPX: Sequential Packet Exchange

SQL: Structured Query Language

SSL: Secure Socket Layer

TGS: Ticket-Granting Service

TGT: Ticket-Granting Ticket

TI: Technologic of Information

TLS: Transaction Layer Security

VPN: Virtual Private Network

W3C: World Wide Web Consortium

XDR: External Data Representation

XNS: Xerox Network Service

Referências

[01] - AMARAL, Wilson Henriques do, "Arquitetura Cliente/Servidor Orientada a Objeto", Tese de Mestrado, DES-IME - Departamento de Engenharia de Sistemas, Instituto Militar de Engenharia, 1993.

[02] - ANDREWS, Gregory R., "*Concurrent Programming: Principles and Practice*", Addison-Wesley, 1991.

[03] - "*Reverse Proxy Documentation*". Disponível em: http://www.apache.org. Acesso em: 10/06/2004

[04] - BATTISTI, Julio, "*Windows Server 2003*: Curso Completo", Editora Axcel Books, 2003

[05] - LEVACHOF, Alessandro, "Uma proposta de extensão da camada de transporte (TLS) para uso sobre datagramas de usuários (UDP)", Tese de Mestrado, UFRJ, 2004

[06] - COMER, Douglas E. & STEVENS, David L. "*Internetworking With TCP/IP Vol.III: Client/Server Programming and Application (Socket Version)*". Prentice Hall, Englewood Cliffs, New Jersey, USA, 1993.

[07] - SOFIA, Helena Rute E. C., "Estudo do Protocolo *ISAKMP/Oakley* como Norma de Gestão de Chaves da Arquitetura de Segurança *IPSec*". Tese de Mestrado, FCUL - Faculdade de Ciências da Universidade de Lisboa/ Departamento de Informática, 1999.

[08] - DANSEGLIO, Mike. MOWERS, David and CRALL, Chris. "*SSL/TLS in Windows Server 2003*". Disponível em: http://technet.microsoft.com. Acesso em: 01/07/2003.

[09] - CUSTER, Helen. "*Windows NT*", *Microsoft Press*, Makron Books, São Paulo, 1993.

[10] - DAVIS, Ralph. "*Windows NT Networking Programming*" Addison Wesley, 1994

[11] - DUMAS, Arthur "Programando WinSock", Axcel Books, Rio de Janeiro, 1995.

[12] - HULTQUIST, Steve <ssh@vnet.ibm.com> "*FAQ about Client/Server*". Disponível em: http://non.com/news.answers/client-server-faq.html. Acesso em: 03/07/1997.

[13] - KORB, Alexei, "Colaboração Visual: Ferramentas, protocolos e aplicações para a interação remota entre pessoas", Tese de Mestrado, UFRGS, 2000.

[14] - SCHIFFMAN, Allan M., "*The Secure Sockets Layer Protocol and Applications*", Western Institute of Computer Science, Internet Security Course, 02/08/1996.

[15] - LEFEBVRE, Alain "*L'Architecture Client-Serveur: Aspects techniques, enjeux stratégiques*", Armand Colin, 2e édition, 1994.

[16] - LOCHER, L. J., "*Trusted and Trusting Domains in NT 4.0*", *Windows* & .NET Magazine", December 1999.

[17] - CARMO, Agnaldo do, *et al*, "Tecnologia da Informação em Governo", Curso de capacitação para executivos públicos, módulo 9, FUNDAP. Disponível em: http://www.fundap.sp.gov.br. Acesso em: 02/06/2003.

[18] - MACHADO, Pedro P. Lemos, BEZERRA, Ernani Lopes, LIMA, José Ney de O., "Fundamentos do Modelo de Segurança da Informação". Disponível em: http://www.redegoverno.gov.br. Acesso em: 10/08/2000.

[19] - FERNANDEZ, Alexandre M., "Segurança em redes IP". Monografia de Pós-Graduação, ASIT (*Advanded School of Internet Tecnology*), 2001.

[20] - STEWART, McKie, "*Everything you wanted to know about Client/Server computing but were afraid to ask*" Eye on Technology. Disponível em: http://www.duke.com/controller/Issues/DecJan/Clientse.htm. Acesso em: 11/07/2002.

[21] - *Microsoft Corporation*, "*Understanding PPTP (Windows NT 4.0)*", *Write Paper* 0197 Part nº 098-68564, 1996. Disponível em: http://technet.microsoft.com. Acesso em: 03/03/2003.

[22] - *Microsoft Corporation*, "*Single Sign-On in Windows 2000 Networks*", *Microsoft paper*,. Disponível em: http://technet.microsoft.com. Acesso em: 03/03/2001.

[23] - *Microsoft Corporation*, "*How VPN Works*". Disponível em: http://technet.microsoft.com. Acesso em: 08/04/2003.

[24] - MURHAMMER, Martin. BOURNE, Tim. GAIDOSCH, Tamas. KUNZINGER, Charles. RADEMACHER, Laura. WEINFURTER, Andreas. "*A Compreensive Guide to Virtual Private Networks*", Volume I: *IBM Firewall, Server and Client Solution*. *Internacional Business Machines Corporation* 1998. Disponível em: http://www.redbooks.ibm.com. Acesso em: 04/05/2003.

[25] - PASTORE, Pablo. Grupo de Teleinformática e Automação-UFRJ. Disponível em: http://www.gta.ufrj.br/grad/03_1/http/tls.html. Acesso em: 30/04/2004.

[26] - PHAM, Thuam Q. & Garg, Pankaj K. "*Multithreaded Programming with Windows NT*" Prentice Hall, 1996.

[27] - QUINN, Bob & Shute, Dave "*Windows Sockets Network Programming*" Addison Wesley, 1996

[28] - RENAUD, Paul E. "Introdução aos Sistemas Cliente/Servidor" IBPI Press, RJ 1993.

[29] - REZENDE, Edmar Roberto S. de, "Segurança no acesso remoto VPN", Dissertação de Mestrado, IC-UNICAMP Instituto de Computação, Universidade Estadual de Campinas, 2004

[30] - ROBERTS, Dave. "*Developing for the Internet with Winsock*" Coriolis Group Books, 1995.

[31] - SALEMI, Joe. “Guia para Banco de Dados Cliente/Servidor” . IBPI Press, 1993.

[32] - SANTOS, Luiz Carlos dos. “Como funciona a autenticação”, artigo publicado em agosto/2000, Clube das Redes. Disponível em: http://www.clubedasredes.eti.br/rede0008.htm. Acesso em 07/06/2004.

[33] - SENA, Jansen C. “Um modelo para proteção do tráfego de serviços baseado em níveis de segurança”. Tese de Mestrado, IC-UNICAMP Instituto de Computação, Universidade Estadual de Campinas, 2002

[34] - SMITH, Randy Franklin. “*Windows 2000 Security Gains*”, *Windows 2000 Magazine*,. Disponível em: http://www.win2000mag.net/, 2001 e http://technet.microsoft.com, 2002. Acesso em: 04/03/2003.

[35] - TANENBAUM, Andrew S. “*Distributed Operating Systems*” Prentice Hall, 1995.

[36] - TANENBAUM, Andrew S. “*Computer Networks*” Prentice Hall, *Third Edition*, 1996.

[37] - TAROUCO, Liane Margarida Rockenbach. “Redes de Computadores: Locais e de Longa Distancia”. McGraw-Hill ,São Paulo, 1986.

[38] - VIDAL, Paulo César Salgado. “Modelagem para Arquitetura Cliente/Servidor Orientada a Objeto”. Tese de Mestrado, DES-IME - Departamento de Engenharia de Sistemas, Instituto Militar de Engenharia, 1994.

[39] - BÜLENT, Yener. “*Internet Security*”, *Rensselaer Polytechnic University, New York*, 2002.

[40] - STEVENS, Richard W., “*TCP/IP Illustrated, Volume 1 the protocols*”, Addison Wesley, 2001

[41] - *Microsoft Corporation*, “*Virtual Private Networks for Windows Server 2003*”. Disponível em: http://technet.microsoft.com. Acesso em: 09/06/2004.

[42] - *Microsoft Corporation, “Cryptography for network and information security*”, *Windows 2000 Server Resource Kit Documentation.* Disponível em: http://technet.microsoft.com. Acesso em: 09/06/2004.

[43] - *Microsoft Corporation*, “*Windows 2000 Certificate Services and Public Key Infrastructure*”, *Windows 2000 Server Resource Kit Documentation*. Disponível em: http://technet.microsoft.com. Acesso em: 10/06/2004.

[44] – Kurose, James F., Ross, Keith W., “*Computer Networking: A TopDown Approach Featuring the Internet*”, Addison Wesley, 3rd Edition

[45] – *Microsoft Corporation*, “*Chapter 1 - Windows NT Networking Architecture*”, *Windows NT Server Documentation*. Disponível em: http://www.microsoft.com/technet/prodtechnol/winntas/support/chptr1.mspx. Acesso em: 27/10/2004.