UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Um Monitor de Transações de Serviços Internet

por

ALEX FABIO PELLIN

Dissertação submetida à avaliação, como requisito parcial
para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. Raul Fernando Weber
Orientador

Porto Alegre, outubro de 2002.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Pellin, Alex Fabio

Um Monitor de Transações de Serviços Internet / por Alex Fabio Pellin - Porto Alegre: PPGC da UFRGS, 2002.

111 f.:il.

Dissertação (mestrado) - Universidade Federal do Rio Grande do Sul. Programa de Pós Graduação em Computação, Porto Alegre, RS – Brasil, 2002. Orientador: Weber, Raul Fernando.

1. Redes de Computadores 2. Segurança de Rede de Computadores I. Weber, Raul Fernando. II. Título.

Agradecimentos

Agradeço especialmente ao Prof. Raul Fernando Weber, que por muitas vezes, além de orientador foi um grande incentivador. A ele, que sempre foi para mim uma grande referência na área, e muito contribuiu para o conhecimento por mim adquirido no decorrer do trabalho, meu agradecimento.

A todos os que me cercaram nestes últimos tempos e, de uma forma ou outra, me apoiaram. São colegas de trabalho, de curso, de final de semana, amigos enfim.

À UCS e à UFRGS que foram as instituições que possibilitaram a realização deste curso.

Obviamente, como não poderia deixar de ser, à minha família, sem exceções. A todos o meu reconhecimento por tudo. Vocês sabem do que eu estou falando.

Quanto mais conhecimento um indivíduo acumula, mais próximo da simplicidade ele está.

Autor desconhecido.

Sumário

Lista de Abreviaturas **7**

Lista de Figuras **8**

Lista de Tabelas **10**

Resumo **11**

Abstract **12**

1 Introdução **13**

2 A Rede Ethernet **15**

2.1 O Modelo de Camadas OSI **16**
2.2 O padrão Ethernet e as Camadas OSI **17**
2.3 Elementos básicos do Ethernet **18**
2.3.1 O quadro Ethernet 19
2.3.2 O protocolo de controle de acesso ao meio 20
2.3.3 Componentes de sinalização 20
2.3.4 Meio Físico 21
2.4 Transporte de Dados nos Quadros Ethernet **22**
2.5 Domínios de Colisão **23**
2.6 Domínios de *Broadcast* e *Multicast* **26**

3 O Conjunto TCP/IP **27**

3.1 O Modelo de Camadas TCP/IP **28**
3.1.1 Camada de Interface com a Rede 29
3.1.2 Camada Internet 29
3.1.3 Camada de Transporte 29
3.1.4 Camada de Aplicação 30
3.2 IP – *Internet Protocol* **31**
3.2.1 Roteamento dos Datagramas 35
3.2.2 Tamanho do Datagrama e Fragmentação 35
3.3 ICMP - *Internet Control Message Protocol* **36**
3.3.1 Mensagens de Eco 38
3.3.2 Destino Inacessível 38
3.3.3 Solicitação de Redução de Índice de Transmissão 39
3.3.4 Solicitação de Redirecionamento 40
3.3.5 Tempo de Vida de um Datagrama Excedido 41
3.3.6 Problemas de Parâmetros de um Datagrama 41
3.3.7 Solicitação de Indicação de Hora 42
3.3.8 Solicitação e Resposta de Informação 43
3.3.9 Obtenção de Máscara de Endereçamento 43

3.4 TCP - *Transmission Control Protocol* 44
3.4.1 Abertura de Conexões 46
3.4.2 Encerramento de Conexões 47
3.4.3 Transmissão e Recepção de Pacotes 48
3.4.4 Retransmissão e Duplicação de Datagramas 49
3.4.5 Utilização de Janelas Deslizantes e Perda de Pacotes 51
3.4.6 Dados Urgentes 55
3.4.7 Controle de Fluxo e Congestionamento 55
3.5 UDP - *User Datagram Protocol* 55
3.6 Portas de Serviço 56
3.7 Cuidados com o uso do TCP/IP 57
3.7.1 Métodos Criptográficos 58
3.7.2 Mecanismos de Autenticação 58
3.7.3 Wrappers 58
3.7.4 Firewalls 59
3.7.5 Detecção de Falhas 60

4 Monitoramento da Rede Ethernet 62

4.1 Para verificar a eficiência do *firewall* 62
4.2 Para orientar a reconfiguração automática de um *firewall* 63
4.3 Monitor para Sistema de Detecção de Intrusão 63
4.4 Monitor para auxiliar na tomada de decisões 64
4.5 Formas de monitorar a rede 65
4.5.1 Monitoramento de pacotes por passagem 65
4.5.2 Monitoramento em Modo Promíscuo 66
4.6 Softwares para o monitoramento 68
4.6.1 darkstat 70
4.6.2 ntop 70
4.6.3 tcpdump 70
4.6.4 snort 70
4.6.5 sniffit 71
4.6.6 ettercap 71
4.6.7 angst 72
4.6.8 ethereal 72
4.6.9 etherape 72
4.6.10 tcpflow 72
4.6.11 ipfm 72
4.6.12 ipaudit 73
4.6.13 iptraf 73
4.7 A Captura na Rede 73
4.7.1 Facilidades Oferecidas pela Libpcap 74
4.7.2 Principais funções da Libpcap 75

5 O Monitor de Transações Internet (MONITIN) 79

5.1 Definição da estrutura de hardware necessária 79
5.2 Definição dos recursos de software exigidos 79
5.3 Construção do algoritmo de monitoramento das conexões 80
5.4 Construção do protótipo para provação 83

6 Validação do Protótipo 88

6.1 Testando a captura de pacotes 89
6.2 Validando o algoritmo de monitoramento de conexões 92
6.2.1 Primeira etapa 92
6.2.2 Segunda etapa 94

7 Conclusão 96

Anexo 1 Tráfego durante a validação do mecanismo de captura. 100

Anexo 2 Tráfego durante a validação do algoritmo - Etapa 1 103

Anexo 3 Tráfego durante a validação do algoritmo - Etapa 2 105

Bibliografia 107

Lista de Abreviaturas

ARP	Address Resolution Protocol
ARPA	Advanced Research Projects Agency
BSD	Berkeley Software Distribution
BPF	BSD Packet Filter
CRC	Cyclic Redundancy Check
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
DIX	Padrão Ethernet criado pelo consórcio DEC-Intel-Xerox
DNS	Domain Name Service
FCS	Frame Check Sequence
FTP	File Transfer Protocol
HTTP	Hipertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Group Management
IP	Internet Protocol
ISO	International Standards Organization
LAN	Local Area Network
LLC	Logical Link Control
MAC	Media Access Control
MONITIN	Monitor de Transações Internet
MR-OSI	Modelo de Referência OSI
MTU	Maximum Transmission Unit
NFS	Network File System
NNTP	Network News Transfer Protocol
NTP	Network Time Protocol
OSI	Open System Interconnect
POP-3	Post Office Protocol – Version 3
RIP	Routing Information Protocol
SDI	Sistema de Detecção de Intrusão
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Monitoring Protocol
TCP	Transport Control Protocol
TELNET	Network Terminal Protocol
TFTP	Trivial File Transfer Protocol
UDP	User Datagram Protoco
VLAN	Virtual LAN

Lista de Figuras

FIGURA 2.1 - Camadas do Modelo MR-OSI 16
FIGURA 2.2 - As principais subcamadas do IEEE 18
FIGURA 2.3 - Quadro Ethernet 19
FIGURA 2.4 - Segmento Ethernet com cabo coaxial 21
FIGURA 2.5 - Segmentos Ethernet com par trançado unidos por repetidor 21
FIGURA 2.6 - Rede estruturada 22
FIGURA 2.7 - *Hub* repetidor une segmentos em único domínio de colisão 24
FIGURA 2.8 - *Hub* de comutação cria domínios de colisão separados 25
FIGURA 2.9 - Comparativo de funcionamento entre *Hub* e o *Switch* 26
FIGURA 3.1 - Camadas do Modelo TCP/IP 28
FIGURA 3.2 - Exemplo de protocolos das camadas TCP/IP 31
FIGURA 3.3 - Cabeçalho IP 32
FIGURA 3.4 - Formato do campo Tipo de Serviço 33
FIGURA 3.5 - Flags do cabeçalho IP 34
FIGURA 3.6 - Encapsulamento de uma mensagem ICMP 37
FIGURA 3.7 - Mensagem ICMP de eco 38
FIGURA 3.8 - Mensagem ICMP de destino inacessível 38
FIGURA 3.9 - Mensagem ICMP de redução de índice de transmissão 39
FIGURA 3.10 - Mensagem ICMP de redirecionamento 40
FIGURA 3.11 - Mensagem ICMP de tempo de vida do datagrama excedido 41
FIGURA 3.12 - Mensagem ICMP de problemas de parâmetros de um datagrama 41
FIGURA 3.13 - Mensagem ICMP de indicação de hora 42
FIGURA 3.14 - Mensagem ICMP de obtenção de máscara de endereçamento 43
FIGURA 3.15 - Cabeçalho TCP 44
FIGURA 3.16 - Estabelecendo uma conexão 47
FIGURA 3.17 - Encerrando uma conexão 48
FIGURA 3.18 - Troca de datagramas entre *host* A e *host* B 49
FIGURA 3.19 - Retransmissão de pacote perdido 50
FIGURA 3.20 - Duplicação de pacote no receptor 51
FIGURA 3.21 - O uso de janelas deslizantes 52
FIGURA 3.22 - Representação da janela deslizante 53
FIGURA 3.23 - Janela deslizante de tamanho 6 54
FIGURA 3.24 - Cabeçalho UDP 56
FIGURA 4.1 - Monitoramento de pacotes por passagem 65
FIGURA 4.2 - Monitoramento em modo promíscuo 67
FIGURA 4.3 - Acessando o pacote memória. 78
FIGURA 5.1 - Algoritmo utilizado para monitorar conexões 82
FIGURA 5.2 - Exibição do contador de pacotes 84
FIGURA 5.3 - Exibição dos campos TCP/IP selecionados 85
FIGURA 5.4 - Contabilização do tráfego, ordenada pelo IP destino. 86
FIGURA 5.5 - Visualização das conexões ativas 86
FIGURA 5.6 - Formato do Arquivo Param.txt 87
FIGURA 6.1 - Ambiente de testes 88
FIGURA 6.2 - Pacotes capturados em cada medição, com filtro1 91
FIGURA 6.3 - Diferença de pacotes capturados entre Monitin e Tcpdump com filtro1 92
FIGURA 6.4 - Pacotes capturados durante o monitoramento de conexões 95

FIGURA 7.1 - Tráfego durante amostragem 1 100
FIGURA 7.2 - Tráfego durante amostragem 2 100
FIGURA 7.3 - Tráfego durante amostragem 3 101
FIGURA 7.4 - Tráfego durante amostragem 4 101
FIGURA 7.5 - Tráfego durante amostragem 5 102
FIGURA 7.6 - Tráfego durante amostragem 6 102
FIGURA 7.7 - Tráfego durante amostragem 7 103
FIGURA 7.8 - Tráfego durante amostragem 8 103
FIGURA 7.9 - Tráfego durante amostragem 9 104
FIGURA 7.10 - Tráfego durante amostragem 10 104
FIGURA 7.11 - Tráfego durante amostragem 11 105
FIGURA 7.12 - Tráfego durante amostragem 12 105
FIGURA 7.13 - Tráfego durante amostragem 13 106

Lista de Tabelas

TABELA 3.1 - MTUs 35
TABELA 3.2 - Tipos de mensagens ICMP 37
TABELA 3.3 - Portas e serviços 57
TABELA 4.1 - Principais Funções da Libpcap 76
TABELA 6.1 - Descrição dos computadores utilizados para os testes 88
TABELA 6.2 - Número de pacote capturados nas medições com Filtro1 90
TABELA 6.3 - Número de pacote capturados nas medições com Filtro2 90
TABELA 6.4 - Número de pacote capturados nas medições com Filtro3 90
TABELA 6.5 - Medições do monitoramento de conexões - etapa 1 93
TABELA 6.6 - Medições do monitoramento de conexões - etapa 2 94
TABELA 7.1 - Comparativo de Ferramentas de Captura de Pacotes 97

Resumo

Monitorar significa, de forma genérica, acompanhar e avaliar dados fornecidos por aparelhagem técnica. Quando se fala em monitoramento de uma rede, não se está fugindo desta idéia. Para monitorar a rede são utilizados mecanismos para coletar dados da mesma, sendo estes dados posteriormente avaliados.

O monitoramento da rede é, sob o ponto de vista da administração da mesma, uma atividade indispensável. Através desta operação é possível obter conclusões sobre a "saúde" da rede. A busca e análise dos dados da rede podem ser feitas com vários enfoques, cada um buscando cercar uma situação específica, onde entre outros, destacam-se a segurança e a carga da rede.

A proposta de fazer uso de algum recurso que permita monitorar a rede fica cada vez mais importante, à medida que as redes têm crescido em tamanho e importância para as organizações. Atualmente, é comum se falar em redes locais com centenas e até milhares de computadores conectados. Associada a esta realidade existe ainda a conexão com a Internet, que faz com que o número de máquinas em contato, suba para valores gigantescos. Os usuários de computador que estão conectados a uma rede, podem estar, fisicamente, muito longe dos olhos do administrador da mesma. Com isso, este sente-se obrigado a utilizar ferramentas que permita monitorar a rede, uma vez que não tem controle sobre os usuários.

Sob o ponto de vista da segurança, a preocupação está em verificar a possível ocorrência de ataques ou detectar problemas nas configurações dos mecanismos de segurança implementados. Já quanto à carga da rede, o enfoque é monitorar os tipos de acessos e serviços utilizados, a fim de identificar atividades supérfluas que possam estar sobrecarregando a rede.

O presente trabalho tem por objetivo estudar meios para construir uma ferramenta que permita verificar, de forma on-line, as conexões TCP/IP que estão ativas na rede local, seja uma conexão entre duas máquinas da rede local, ou com a Internet, possibilitando visualizar os serviços que estão sendo acessados e a quantidade de tráfego gerada pelos computadores. Ao final será construído um protótipo a fim de validar o estudo feito.

O estudo parte da análise do padrão de rede Ethernet, que é ambiente a ser utilizado neste estudo. Na seqüência serão estudadas as características dos principais protocolos da família TCP/IP, que é o conjunto de protocolo utilizado pela grande maioria das redes, inclusive pela maior delas, que é a Internet.

Em uma fase posterior, serão estudadas as formas de se fazer o monitoramento em uma rede Ethernet e as ferramentas de monitoramento existentes.

Na seqüência, os detalhes do protótipo para monitorar conexões TCP/IP são apresentados bem como os resultados dos testes de validação do mesmo.

Palavras-chave: Ethernet, TCP/IP, Monitoramento de Rede, Captura de Pacotes

TITLE: “AN INTERNET SERVICES TRANSACTION MONITOR”

Abstract

Monitoring, in general terms, means following and evaluating data supplied by technical equipment. When we talk about network monitoring, we are not far from this. To monitor the network we must use some mechanisms to collect data, and then evaluate them afterwards.

Network monitoring is an indispensable activity under a administration point-of-view. Through this operation it is possible to achieve conclusions on the network “health”. The search and analysis of network information can be done under several perspectives, each one trying to focalize an specific situation, from which we could detach network loading and security.

As networks have grown in size and importance for organizations, the proposal of using any resource that allowed the net monitoring gets more and more important. Nowadays it is common to speak of local networks with hundreds or even thousands of computers connected. Besides, there is the Internet connection, which makes the number of machines interconnected reach giant proportions. Users connected to a computer network may be physically far away from its administrator, therefore, this one must use tools which can monitor the network, once he can not control the users.

Under the security point-of-view, the concern is to check possible strike occurrences or to detect problems on configurations of implemented security mechanisms. On other hand, the focus on the network loading is to monitor the types of accesses and services utilized, in order to identify useless activities which might be overloading the network.

This current work aims to study means of building a tool which can verify, on-line, active TCP/IP connections on the local network, either within two machines in the local network or with Internet, allowing the viewing of services being accessed and the volume of traffic generated by computers. At the end, a prototype shall be constructed in order to validate the theorical work.

The work starts analyzing the Ethernet standard, which is the environment this work is based on. Afterwards, the features of TCP/IP main protocols, which is the set of protocols utilized by most nets (including the largest: Internet) shall be studied.

In a posterior phase, ways of doing the monitoring on an Ethernet network and the existent tools of monitoring will be studied.

Details of the prototype to monitor TCP/IP connections are presented next, as well as the results of its validation tests.

Keywords: Ethernet, TCP/IP, Network Monitoring, Packets Capture

1 Introdução

Não é nenhuma novidade afirmar que as redes de computadores estão cada vez maiores, interligando internamente, os diversos departamentos das organizações e estas, umas com as outras, através da Internet.

Para fazer o controle da segurança das redes, são definidas políticas de segurança que envolvem alguns aspectos de uso dos recursos da rede [BRE 99]. A partir das políticas, são implementados os mecanismos que regem a segurança e o uso da rede, conforme o que foi definido.

Sob o aspecto da segurança, a implementação de *firewalls* [CHA 95], é o método mais utilizado para controlar o acesso entre duas redes. Sua função é proteger a rede local (rede interna) de acessos indevidos, originados externamente.

Além de acessos que correspondam a ataques, o *firewall* pode ser utilizado para bloquear certos tipos de tráfego, que mesmo sendo inofensivos, são considerados supérfluos. Pode ser chamado de supérfluo, por exemplo, o tráfego de rede que não tem relação com a atividade principal da organização.

Como pode se observar, a implementação de mecanismos de *firewall*, é útil para as organizações. Porém, não estão livres de falhas de implementação ou configuração. É possível que as configurações feitas no mecanismo de segurança não reflitam exatamente o que foi definido nas políticas de segurança.

Para verificar a situação, devem ser utilizadas ferramentas específicas. Um monitor de rede, conforme proposto neste estudo, tem esta característica, pois permite que se compare o que realmente está trafegando na rede, com o que é definido, nas políticas de segurança, como tráfego permitido. Um monitor de rede também pode ser utilizado para que o administrador da rede identifique a ocorrência, por exemplo, de novos tipos de tráfego supérfluo. Possibilitando, desta forma, que seja executado um processo de reconfiguração do mecanismo de segurança, a fim de eliminar tal ocorrência.

Para desenvolver este trabalho, é feito um estudo da tecnologia de rede Ethernet. Esta escolha se deve ao fato de que a rede Ethernet é muito utilizada em redes locais, que é o enfoque do monitoramento a ser executado [SPU 2000]. O estudo feito sobre esta tecnologia permite compreender como se é feita a comunicação entre os computadores conectados, possibilitando identificar meios de fazer a obtenção dos dados relativos ao tráfego.

O estudo do TCP/IP, que compreende um grupo de protocolos amplamente utilizados nas redes locais e é a base da comunicação na Internet, serve para identificar as características dos principais protocolos relacionados, que são: IP, TCP, UDP e ICMP [TAN 97] [COM 98].

Existem ferramentas de monitoramento disponíveis, porém algumas com propósitos diferenciados, não contemplam as características de monitorar as conexões de forma *on-line*, ou então, não exibem a quantidade de tráfego que está associada a cada conexão. Outras são voltadas para as atividades de ataques, onde que a sua concepção não se importa com a performance ou segurança da rede [HAT 2002].

Apesar deste estudo utilizar recursos de captura de pacotes, o que poderia ser considerado algo ilícito, ele é voltado para o lado bom da informática. Isto é, objetiva construir uma ferramenta que auxilie o grupo de administração da rede. Não como um sistema de detecção de intrusão por si só, mas como parte de um conjunto de ferramentas que podem ser utilizadas para auxiliar na administração da segurança da

rede. A ferramenta proposta, mediante a interação com o usuário, deverá permitir focalizar uma parte específica do trafego da rede, para análise *on-line*. Ela possibilitará o monitoramento das conexões ativas sobre o protocolo internet, exibindo informações relativas a cada uma. Também deverá gerar informações sobre o fluxo de dados existente entre os *hosts* da rede.

O presente texto está dividido em sete capítulos. No capítulo 2 está um estudo do padrão Ethernet, enquanto o capítulo 3 aborda o suíte TCP/IP, aprofundando os principais protocolos associados. O monitoramento de redes Ethernet é tratado no capítulo 4. Nele, são apresentadas alternativas para realização de monitoramento. Também são estudadas algumas ferramentas para monitoramento da rede. No capítulo 5, tem-se o registro da construção da ferramenta proposta, que tem o relato da validação do seu protótipo no capítulo 6. O capítulo 7 apresenta as conclusões e comentários sobre possíveis futuros trabalhos.

2 A Rede Ethernet

Ethernet é a tecnologia de rede local mais utilizada no mundo, segundo [SPU 2000]. Esta tecnologia, que está próxima de completar três décadas de existência, passou por muitas inovações até se tornar “a mais popular do mundo”.

Em maio de 1973, Bob Metcalfe (funcionário da Xerox) descreveu um sistema para interconexão de estações de trabalho de computador, era o surgimento do Ethernet.. Tal sistema possibilitou o envio de dados entre si e para impressoras laser de alta velocidade.

O estudo de Metcalfe se desenvolveu a partir de experiências anteriores com a rede Aloha, que foi iniciada nos anos 60. O protocolo Aloha é simples, cada estação pode enviar o que quiser, quando quiser, e depois aguardar a confirmação de recebimento por parte do destinatário. Como transmissões combinadas são desprezadas e não retornam confirmação, então o emissor aguarda um determinado tempo. Se a confirmação não for recebida, assume que houve uma colisão e aguarda um tempo aleatório para a retransmissão. Para redes de pouco tráfego, este protocolo era aceitável.

Para melhorar esta técnica, foi incluído o uso de *slots* para a transmissão, utilizando um relógio mestre para sincronização.

O sistema de Metcalfe incluía um mecanismo de detecção de colisão (*collision detection*) e o conceito de “ouvir” o sinal da portadora antes de “falar” (*carrier sense*). Isso tudo fazendo uso de um meio compartilhado, de acesso múltiplo (*multiple access*). Baseado nesses conceitos vem o nome do protocolo de acesso ao meio Ethernet, o CSMA/CD (Carrier Sense Multiple Access with Collision Detection).

A rede experimental trabalhou a 2,94 Mbps. Em 1980, o consórcio de empresas DEC-Intel-Xerox publicou o padrão Ethernet para 10 Mbps, que foi conhecido como padrão DIX (iniciais das empresas consorciadas).

Quando o padrão DIX foi publicado, o IEEE (*Institute of Electrical and Electronics Engineers*) já fazia estudos em busca de um padrão de redes abertas. Então em 1985, o IEEE assumiu com algumas inovações a fim de atender a padronização mundial de redes LAN (*Local Area Network*), o sistema de rede descrito no padrão DIX original, publicando o padrão IEEE 802.3, que ficou sendo o padrão Ethernet oficial.

O grande crescimento do uso do Ethernet se deu no final dos anos 80, quando foi inovado o padrão, para trabalhar com par trançado. Esse cabeamento oferece maior facilidades para instalação, gerenciamento e diagnóstico de falhas do que o cabo coaxial, originalmente empregado.

Em 1995, com a adoção do padrão Fast Ethernet de 100 Mbps, a velocidade da rede foi multiplicada em 10 vezes. Tal inovação já se fazia necessária, em função do avanço na tecnologia de computadores, pois o padrão de 10 Mbps já deixava a desejar mesmo para os computadores mais comuns. Este novo padrão permite o uso de fibra ótica e par trançado.

Também sobre estes dois meios físicos foi descrito o padrão Gigabit Ethernet em 1998, que trabalha a 1 bilhão de bits por segundo.

Segundo o padrão, as interfaces de mais alta velocidade, possuem a característica de velocidade múltipla, podendo operar nas velocidades inferiores, fazendo uso de um protocolo de negociação para a sua configuração.

Para poder-se compreender melhor e aprofundar o assunto, deve-se estudar o modelo de camadas de rede. A seguir, tem-se o modelo que é utilizado como referência e auxilia o processo didático da área de redes, O MR-OSI (Modelo de Referência OSI).

2.1 O Modelo de Camadas OSI

O Modelo de Referência OSI (*Open System Interconnect*) criado pela ISO (*International Standards Organization*) possui 7 camadas (Figura 2.1).

FIGURA 2.1 - Camadas do Modelo MR-OSI

As camadas são divisões feitas sobre o problema da comunicação de dados. A divisão do problema em partes menores permite que sejam criados protocolos menores e mais simples para cada uma delas, ao invés de criar-se um protocolo gigantesco. A esse grupo de protocolos dá-se o nome de conjunto ou família. Ao final, o somatório de todos os protocolos de um conjunto, resolve o problema da comunicação com um todo.

A camada define as funções específicas dos protocolos de comunicação de dados, necessárias para garantir que as aplicações se comuniquem através da rede. Não existe regra quanto ao número de protocolos existentes em cada camada, uma camada pode abrigar múltiplos protocolos.

Abaixo tem-se a descrição das principais funcionalidades de cada um dos níveis do modelo:

Aplicação: É o nível mais alto da hierarquia. Aqui residem os programas de aplicação que fazem uso dos recursos da rede, os processos de rede que são acessados diretamente pelo usuário; A camada de Aplicação faz a interface entre o protocolo de comunicação e o aplicativo que pediu ou receberá a informação através da rede;

Apresentação: Para se comunicar, as aplicações necessitam entrar em acordo sobre como os dados serão representados. Esta camada oferece padrões para a apresentação de dados para as aplicações;

Sessão: A camada de sessão permite que duas aplicações em diferentes máquinas estabeleçam uma sessão de comunicação.

Transporte: Oferece a detecção e correção de erros fim a fim, garantindo que o destinatário receba os dados exatamente como foram enviados;

Rede: Faz o gerenciamento das conexões através da rede, mantendo os níveis superiores isolados dos detalhes desta tarefa. A camada de Rede é responsável pelo endereçamento dos pacotes, convertendo os endereços lógicos em endereços físicos, de forma que os pacotes consigam chegar corretamente ao destino. Essa camada também determina a rota que os pacotes irão seguir para atingir o destino, baseada em fatores como condições de tráfego da rede e prioridades;

Enlace: Oferece serviço confiável de entrega de dados, através da rede física; Como a camada física apenas aceita e transmite um fluxo de bits, sem qualquer preocupação em relação ao significado ou à estrutura, cabe à camada de enlace de dados criar e reconhecer os limites do quadro. A principal tarefa da camada de enlace de dados é transformar um canal de transmissão bruta de dados em uma linha que pareça livre dos erros de transmissão não detectados na camada de rede;

Física: A camada física pega a os quadros enviados pela camada de enlace e os transforma em sinais compatíveis com o meio, onde os dados deverão ser transmitidos. Define as características da comunicação sobre o meio físico da rede. Isto é, como os dados serão transportados, quais serão os níveis de voltagens para representar os sinais, pinagem a ser utilizada, são detalhes descritos aqui.

Nessa estrutura, na entidade transmissora, os dados enviados pela aplicação descem pela pilha, passando por todas as camadas até atingir o meio físico. Em cada camada, novas informações de controle são adicionadas aos dados originais. Uma vez colocado no meio físico, os dados trafegam até o destino. Após a chegada na entidade receptora, os dados fazem o caminho inverso, subindo a pilha, até atingir a aplicação destino [SOA 95] [TAN 97]. Em cada camada as informações de controle adicionadas na entidade transmissora são respectivamente retiradas, restando para o nível de aplicação, somente os dados originais.

Tendo sido vistas as características principais do modelo OSI, pode-se agora verificar a relação dos padrões Ethernet com o mesmo.

2.2 O padrão Ethernet e as Camadas OSI

O padrão Ethernet envolve elementos das camadas física e de enlace (*link* de dados) do modelo OSI. Cada uma dessas duas camadas é dividida em subcamadas, conforme pode ser visualizado na Figura 2.2 a seguir:

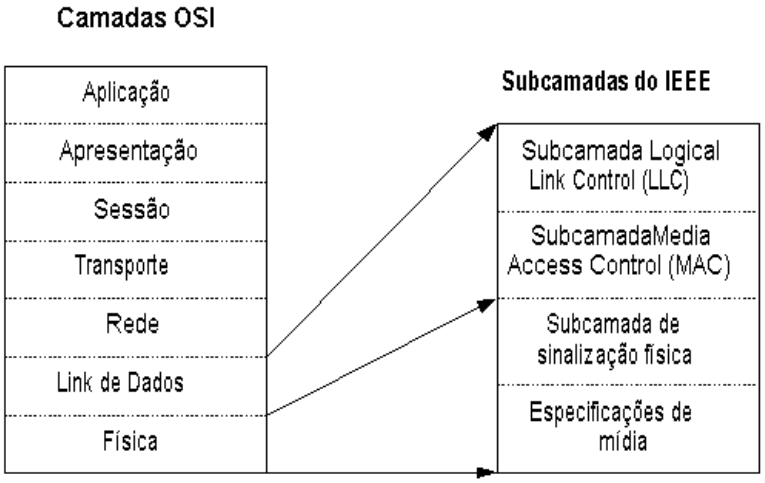

FIGURA 2.2 - As principais subcamadas do IEEE

No nível *link* de dados, existem as subcamadas de controle lógico de *link*, o LLC (*Logical Link Control*) e MAC (*Media Access Control*) que são iguais para todas as variações e velocidades do padrão Ethernet.

A camada LLC foi definida pelo IEEE para identificar os dados transportados em um quadro Ethernet. Fica a encargo da camada MAC a definição do protocolo para fazer o controle do acesso ao sistema Ethernet.

O padrão LLC do IEEE é independente do padrão de LAN Ethernet 802.3, não mudando, seja qual for o sistema LAN utilizado. Por não ser específico do Ethernet, não é formalmente especificado pelo sistema IEEE 802.3. Porém, todas as camadas abaixo da subcamada LLC são exclusivas de cada tecnologia LAN em questão, neste caso o Ethernet.

O modo *full-duplex* não exige protocolo MAC, mas no caso de transmissões *half-duplex*, é ele que permite que um conjunto de estações dispute o acesso ao canal Ethernet compartilhado, de forma justa e equilibrada. Como nesse padrão de LAN não existe um controlador central, cada interface opera de forma independente, usando o mesmo protocolo MAC, que é baseado no CSMA/CD.

No nível físico, as subcamadas do IEEE variam conforme o tipo de meio utilizado e também de acordo com o sistema Ethernet, seja a 10, 100 ou 1000 Mbps.

2.3 Elementos básicos do Ethernet

Um sistema Ethernet é resultante da combinação de quatro blocos de montagem [SPU 2000]:

- O quadro ou *frame*;
- O protocolo de controle de acesso ao meio;
- Os componentes de sinalização;
- O meio físico.

2.3.1 O quadro Ethernet

O quadro é um conjunto padronizado de bits, usados para transportar dados pelo sistema. É na verdade, o elemento mais importante do sistema Ethernet. Toda a estrutura física da rede existe para permitir o deslocamento do quadro entre os computadores que estão conectados na mesma.

A construção do quadro Ethernet respeita uma estrutura de bits, divididos em campos específicos. A Figura 2.3 abaixo representa um quadro Ethernet:

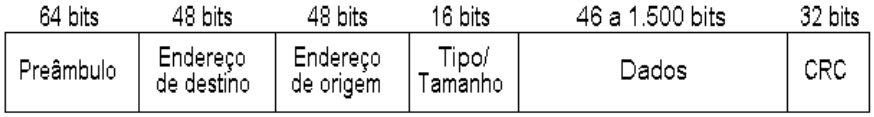

FIGURA 2.3 - Quadro Ethernet

- **Preâmbulo**: Tem a função de gerar uma sinalização ao hardware e aos meios eletrônicos do sistema Ethernet 10 Mbps. Essa sinalização tem a função de fazer com que seja reconhecida a transmissão de um quadro, indicando que dados estarão sendo recebidos na seqüência. Sistemas de 100 e 1000 Mbps não necessitam do preâmbulo, pois a sinalização é constante, mas por questões de compatibilidade, o quadro não muda e o campo permanece.
- **Endereços de Destino e Origem**: Este endereço de 48 bits é também chamado de endereço de *hardware* (endereço físico) das interfaces Ethernet origem e destino do quadro. Para cada interface construída é atribuído um endereço único. Para tal, cada fabricante deve respeitar uma padronização mundial.
- **Tipo ou Tamanho**: Este campo poderá conter dois tipos de informação. O tipo do quadro ou o tamanho do mesmo. Apesar de parecer ambíguo, a situação é facilmente compreendida. Se o valor do campo for menor ou igual que o tamanho máximo do quadro, 1.518 (sistema decimal), então o campo está sendo utilizado para armazenar informação de tamanho do quadro. Nessa situação o campo indica o número de octetos de dados que estão embutidos no campo seguinte, dentro do quadro. Quando ocorre um preenchimento do campo de dados, na montagem do quadro, a fim de atingir o tamanho mínimo necessário, este campo se faz útil. Tal informação é importante no momento do recebimento do quadro, a fim de determinar o tamanho de dados válidos no campo de dados, descartando os dados de preenchimento. Caso o valor desse campo seja maior ou igual que 1.536 (decimal) ou 0x600 (hexa), então o campo está sendo utilizado para armazenar o tipo do protocolo, dos dados que estão sendo transportados no campo dados do quadro.
- **Campo de Dados**: Este campo possui o tamanho mínimo de 46 bytes, podendo chegar até 1.500. O tamanho mínimo deve ser respeitado para que o sinal do quadro permaneça na rede por um tempo mínimo. Este tempo deve ser suficiente para que cada estação Ethernet conectada na rede consiga

"ouvir" o quadro no tempo correto. Isso é importante para o mecanismo de detecção de sinal e colisão.

- **Campo FCS**: o campo *Frame Check Sequence*, também chamado de CRC (*Cyclic Redundancy Check*), é utilizado para fazer a verificação da integridade do quadro inteiro. Serve para que a interface Ethernet receptora verifique se os bits no quadro permanecem intactos após a viagem pela rede.

Estes são os campos existentes num quadro Ethernet tipicamente usado pela grande maioria das LANs. Porém, um novo campo, de quatro bytes, pode ser inserido nessa estrutura, entre o endereço de origem e o campo tamanho/tipo. Trata-se do cabeçalho VLAN (*Virtual* LAN), definido pelo padrão VLAN IEEE 802.1Q de 1998, a fim de proporcionar a implementação de VLANs independente do fabricante

A função deste campo é identificar a qual VLAN o quadro pertence. As VLANs são implementadas em *hubs* de comutação para direcionar o tráfego Ethernet entre o conjunto portas que estão definidas como membros de um mesmo grupo.

Como os cabeçalhos de marcação de VLAN são incluídos em quadros Ethernet pelos *hubs* de comutação e outros dispositivos que foram programados para enviar e receber quadros desse tipo, esta característica não afeta a operação tradicional do Ethernet.

2.3.2 O protocolo de controle de acesso ao meio

Esse protocolo consiste em um conjunto de regras embutidas nas interfaces Ethernet, a fim de possibilitar que vários computadores acessem o canal compartilhado de forma ordenada. Ele apresenta uma característica básica. Cada computador equipado com Ethernet opera de modo independente de todas as outras estações da rede, não existindo, portanto, um controlador central. O sistema Ethernet utiliza o mecanismo de entrega por *broadcast*. Isto é, cada quadro transmitido é ouvido por cada estação.

Existe um canal compartilhado por todas as estações ligadas na rede Ethernet. Os sinais são transmitidos da interface para este canal. Mas antes de enviar dados, cada estação deve ouvir este canal, se estiver ocioso, então a interface efetivamente envia os dados através de quadros (pacotes).

Este mecanismo de controle de acesso ao meio utiliza o protocolo CSMA/CD (*Carrier Sense Multiple Access / Collision Detection*). Onde cada estação precisa ouvir um período de "silêncio" antes de enviar dados. Quando ocorre tal espaço, a estação poderá iniciar a transmissão. Se mais de uma estação iniciar a transmissão simultaneamente, então todas param novamente e aguardam tempos aleatórios para fazer novas tentativas de transmissão.

2.3.3 Componentes de sinalização

Os componentes de sinalização são dispositivos eletrônicos padronizados que enviam e recebem sinais em um canal Ethernet, incluem interface, transceptor, cuja palavra é originada da união das palavras transmissor e receptor, e o cabo que une o transceptor à interface, quando não estão embutidos.

A interface é a parte eletrônica responsável por formar e enviar quadros, assim como, por receber quadros vindos da rede e extrair os dados.

O transceptor, que pode estar ou não embutido na interface, é a parte eletrônica responsável por pegar os sinais da interface da estação e transmiti-los para o segmento de cabo e, também o inverso, pegar os sinais do cabo e enviá-los a interface.

2.3.4 Meio Físico

O conceito de meio físico ou mídia física engloba além dos cabos, outros *hardwares* utilizados no transporte de sinais digitais entre os computadores da rede.

O tipo de cabo utilizado pode variar. Podem ser utilizados cabos de par trançado, fibra ótica ou até mesmo o cabo coaxial. Com equipamentos de rede específicos, é possível que se monte um único canal de rede, fazendo uso destes diferentes sistemas de mídia.

Quando se faz uso do cabo coaxial, num mesmo segmento, pode-se ter vários computadores ou equipamentos de comunicação conectados a ele, como pode ser visualizado na Figura 2.4. Já os outros cabos, são utilizados para conectar dois equipamentos ou computadores, um em cada extremidade, como ilustrado na Figura 2.5.

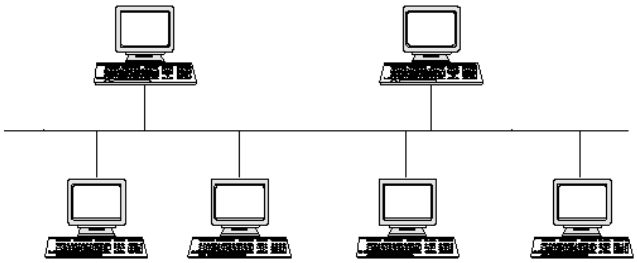

FIGURA 2.4 - Segmento Ethernet com cabo coaxial

Para se ter uma rede Ethernet, basta simplesmente se ter duas estações conectadas num mesmo cabo, ou várias estações conectadas a diferentes segmentos que são unidos por meio de um repetidor, fazendo com que funcionem como um único canal Ethernet compartilhado. A Figura 2.5 representa o uso de repetidores para reunir múltiplos segmentos.

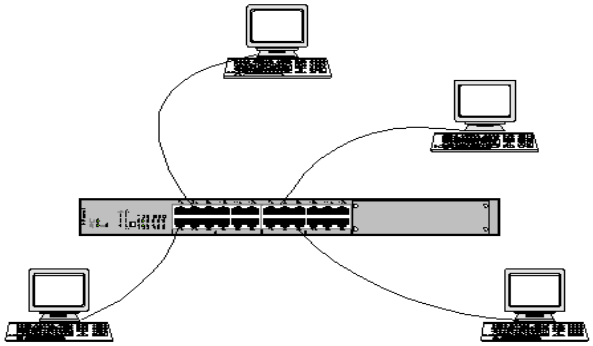

FIGURA 2.5 - Segmentos Ethernet com par trançado unidos por repetidor

Com a união de segmentos, é possível montar estruturas maiores, em forma de árvores, por exemplo. A Figura 2.6 representa esta estruturação da rede.

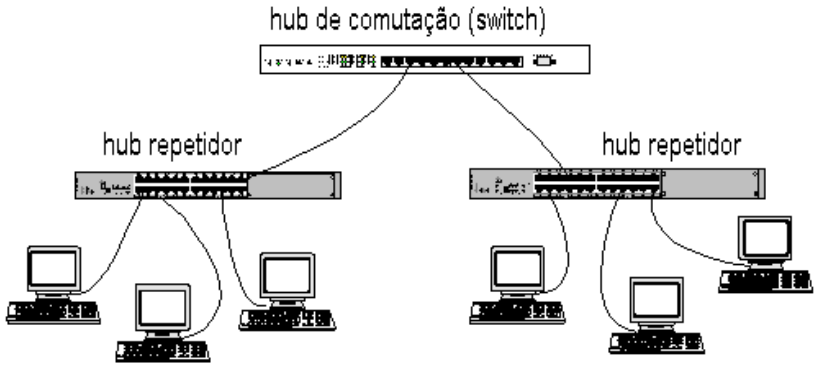

FIGURA 2.6 - Rede estruturada

Tais estruturações ficam limitadas em seu tamanho, em função das características físicas relacionadas ao tempo necessário para que os sinais percorram todo o canal.

Para manter a qualidade faz-se a segmentação de redes maiores. Equipamentos de repetição servem para unir segmentos de cabos em um único canal compartilhado, Figura 1.6. Por meio de equipamentos de segmentação, pode-se segmentar o que é uma única LAN em LANs múltiplas, diminuindo assim a competição pelo uso do meio.

2.4 Transporte de Dados nos Quadros Ethernet

Os dados que trafegam pela rede Ethernet ficam embutidos no campo de dados do quadro e são estruturados conforme protocolos de níveis mais altos, como o conjunto TCP/IP, IPX/SPX, entre outros.

Para a rede Ethernet, o protocolo de alto nível que está sendo utilizado é transparente. Sendo possível ainda, que em uma mesma rede se tenha múltiplos protocolos de alto nível rodando.

Como cada um desses protocolos de nível superior possui uma estrutura diferente, no momento do recebimento de um quadro, a estação destino precisa identificar o protocolo utilizado para poder retirar os dados de forma correta. Para isso usa a informação contida no campo tipo de quadro. Caso este campo contenha o tamanho, então as informações de tipo estarão armazenadas nos primeiros bytes do campo de dados.

Os protocolos de rede de alto nível possuem seu próprio sistema de endereçamento. No caso do IP, que é o mais difundido mundialmente e o objeto principal desse estudo, são utilizados endereços de 32 bits, que são atribuídos às interfaces pelos administradores. Porém o endereçamento utilizado no nível dos quadros Ethernet é o endereço físico da interface, de 48 bits, que é atribuído de forma única, pelo fabricante da mesma.

Numa rede TCP/IP, as aplicações conhecem o endereço IP da máquina com a qual deseja se comunicar, porém para poder montar o quadro a ser enviado, é preciso saber o endereço físico da interface destino.

Para descobrir isso, é feito o uso do protocolo ARP (*Address Resolution Protocol*), cujo funcionamento é simples. Quando uma estação precisa enviar um quadro para um IP, cujo endereço físico não é conhecido, esta envia um pacote *broadcast*, que é estudado mais adiante neste capítulo, para toda a rede, contendo um pedido ARP.

Num pedido ARP, a estação solicita que a estação dentro da rede Ethernet, que possuir o endereço IP especificado, informe o endereço físico da interface correspondente. Ao receber a resposta, a estação solicitante armazena a informação em uma tabela na memória, chamada *cache* de ARP, onde estão associados os IPs e endereços físicos conhecidos.

2.5 Domínios de Colisão

A colisão é detectada pelo transceptor, que identifica a ocorrência simultânea de atividade nos percursos de entrada e saída de dados.

Um domínio de colisão refere a um único sistema Ethernet *full-duplex*, em que os elementos de hardware da rede fazem parte do mesmo domínio de temporização de sinal. Dentro de um domínio de colisão somente uma entidade pode estar transmitindo em determinado instante e todas as entidades podem ouvir umas às outras. Uma colisão ocorre quando dois ou mais dispositivos transmitirem ao mesmo tempo.

O conceito de domínio de colisão é importante para se compreender a utilização de repetidores. Através do uso de equipamentos específicos chamados de *hubs* repetidores ou simplesmente de *hubs*, pode-se unir vários segmentos em um único domínio de colisão, ver Figura 2.7, onde todos os computadores disputam o acesso ao meio físico, onde somente um poderá estar transmitindo num determinado instante, e quando mais de um tenta fazê-lo, ocorre uma colisão.

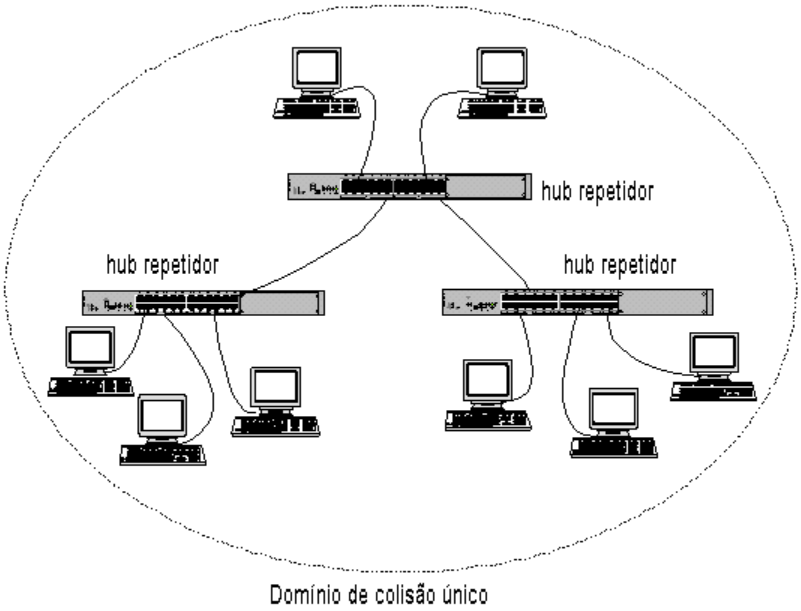

FIGURA 2.7 - *Hub* repetidor une segmentos em único domínio de colisão

O repetidor não toma nenhuma decisão baseada no endereço destino do quadro. Funciona como um amplificador de sinal. É um dispositivo eletrônico analógico que monitora de forma contínua os sinais do cabo, regenerando os sinais recebidos para todas os segmentos conectados, com exceção do segmento de onde o sinal se originou [TOR 00].

Como dentro de um domínio de colisão as estações estão competindo para transmitir sobre um mesmo meio físico, existe um problema em se montar redes grandes com domínio único. A concorrência pelo meio é muito grande.

Para aumentar o número de computadores conectados na rede Ethernet, sem prejudicar a performance da comunicação, são utilizados *hubs* de comutação, também conhecidos como *switchs*. Estes equipamentos têm capacidade de separar domínios de colisão. Isso é, o *switch* isola as máquinas que estão atrás de cada porta em um domínio diferente e é responsável por fazer a comunicação entre eles, conforme pode ser observado na Figura 2.8.

Quando se tiver somente uma estação conectada na porta de um hub de comutação, ela será a única a disputar o acesso ao meio, naquele segmento. Por esta razão, se a porta do *switch* e a interface da estação permitirem, pode-se configurar uma comunicação *full-duplex*, isto é, uma estação poderá enviar e receber dados simultaneamente. Em função desta característica, usualmente são conectados nestas portas, servidores ou estação com missão crítica.

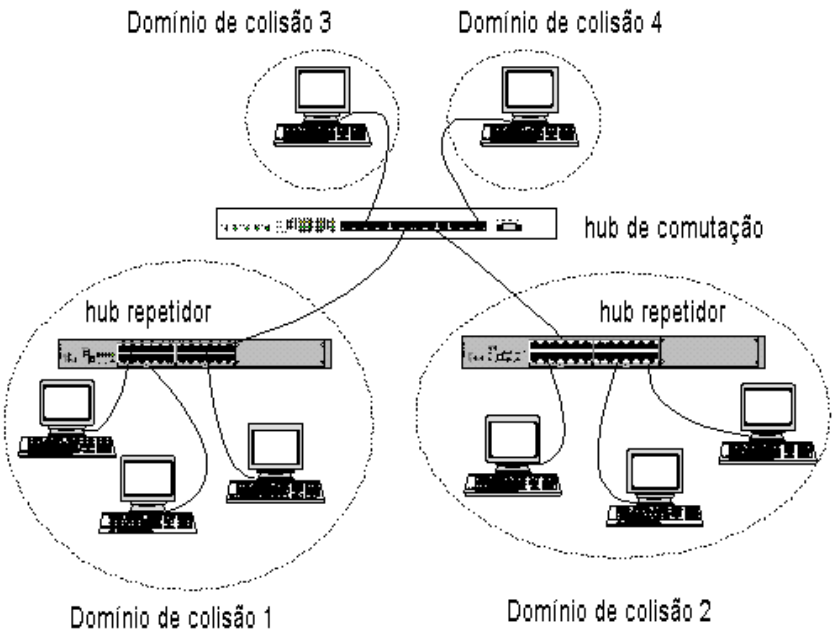

FIGURA 2.8 - *Hub* de comutação cria domínios de colisão separados

De forma diferente de um *hub* repetidor, um *hub* de comutação toma decisões baseadas em endereços de quadros. Um quadro é enviado somente para a porta na qual está o segmento que o destinatário está conectado, seja direta ou indiretamente. Para isso, esse equipamento faz chaveamento entre as portas, permitindo também que mais de uma comunicação seja realizada no mesmo momento. Essa característica pode ser observada na Figura 2.9.

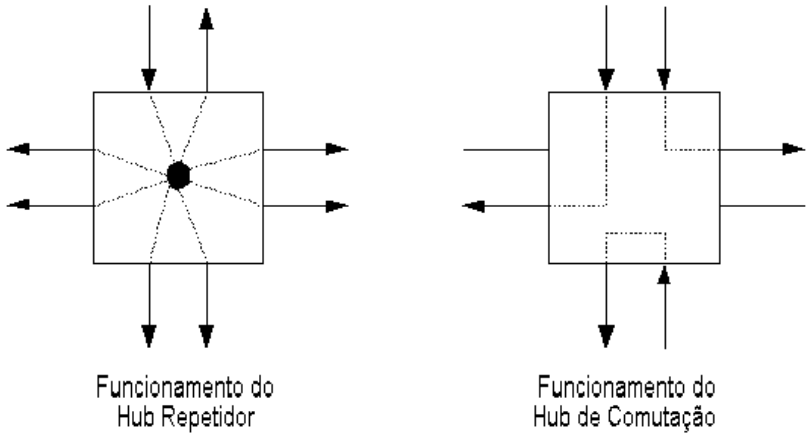

FIGURA 2.9 - Comparativo de funcionamento entre *Hub* e o *Switch*

O derramamento de quadros para todas as portas, como ocorre no *hub* repetidor, ocorre somente quando o *switch* não souber a localização do destinatário. Porém, assim que esta máquina desconhecida, enviar algum quadro que passe pelo equipamento de rede, o *hub* de comutação passará a conhecer a sua localização física, não havendo mais *flooding* para os pacotes a ela direcionados.

2.6 Domínios de *Broadcast* e *Multicast*

Um endereço *multicast* é um endereço que representa um grupo várias interfaces. Quando um quadro é direcionado para este grupo, todas as interfaces pertencentes a ele o receberão. O *broadcast* é um grupo especial de *multicast*. Envolve todas as estações da LAN.

As estações enviam pacotes *multicast* e *broadcast* por diversos motivos. Alguns protocolos utilizam este tipo de quadro como uma forma de descoberta de endereços. Também podem ser utilizados em aplicações específicas, como multimídia, onde as informações de áudio e vídeo são colocadas em quadros desse tipo, a fim de serem distribuídas para múltiplas estações.

Tanto no *hub* repetidor como no *hub* de comutação enviam os pacotes de *broadcast* para todas as portas, exceto para a porta por onde foi recebido o *broadcast*. Desta forma, este tipo de pacote se espalha por toda a LAN.

Para isolar domínios de *broadcast* utiliza-se roteadores, que não encaminham adiante este tipo de pacotes automaticamente.

3 O Conjunto TCP/IP

TCP/IP, que é freqüentemente chamado de protocolo, na verdade identifica um conjunto de protocolos e recebe o nome dos dois principais integrantes. O TCP (*Transport Control Protocol*) é protocolo da camada de transporte, enquanto o IP (*Internet Protocol*) pertence à camada internet do modelo de referência TCP/IP [COM 2001].

O conjunto TCP/IP engloba os protocolos utilizados na grande rede de computadores chamada Internet.

O estudo para desenvolvimento do suíte TCP/IP iniciou nos anos 70. A idéia motivadora foi a ligação inter-redes, possibilitando se conectar redes diferentes, unificando-as. Esse processo se deu ao mesmo tempo em que as redes locais estavam sendo desenvolvidas. Como em tantas outras situações, a pesquisa recebeu apoio da área militar, uma vez que o exército dos EUA foi uma das primeiras organizações a ter múltiplas redes físicas.

A ARPA (*Advanced Research Projects Agency*) foi a grande idealizadora do projeto que inicialmente ligou universidades, e departamentos governamentais e militares americanos. Na época ainda denominada Arpanet, deu origem à Internet[1].

Embora não tenha sido o único conjunto de protocolos a ser desenvolvido para a ligação inter-redes, esse suíte foi o que se firmou e é ainda hoje, o mais utilizado.

Algumas características do TCP/IP influenciaram no crescimento de sua popularidade [HUN 94]:

- Utiliza padrões de protocolo aberto, livremente disponível e desenvolvido de forma independente a qualquer *hardware* ou sistema operacional. Por ser amplamente suportado, o TCP/IP é ideal para comunicar diferentes equipamentos e programas, mesmo que isso não seja feito através da Internet.
- Apresenta independência quanto ao *hardware* físico da rede. Isso é importante para que seja possível interligar diferentes tipos de rede. O TCP/IP pode operar, por exemplo, sobre Ethernet, token-ring e linhas discadas, permitindo a interligação de redes distintas.
- O seu esquema de endereçamento permite que se possa identificar de forma única na rede, qualquer equipamento configurado, independentemente do tamanho da rede.
- Possui protocolos de alto nível padronizados para os muitos serviços de usuários oferecidos.

Com o surgimento da possibilidade de se interligar e unificar as redes distintas, a interconexão se tornou algo essencial para a sobrevivência das organizações. Com isso, ganhou força a Internet, que é a grande rede mundial de computadores que possibilita a troca de informações fácil e rapidamente entre pontos dispersos geograficamente. Esta rede que encurta as distâncias e socializa o conhecimento, atinge a todos de forma direta ou indireta. O que há alguns anos era praticamente para o uso “acadêmico”, se expandiu e atinge hoje todos os tipos de instituições governamentais e privadas.

[1] A palavra “Internet”, com letra inicial maiúscula indica a rede mundial de computadores, caso contrário faz outra referência.

O emprego do TCP/IP na grande rede mundial fez com que sua utilização crescesse de forma exponencial. Atualmente, organizações fazem uso deste suíte de protocolos, seja para se comunicar com outras, ou simplesmente para interligar equipamentos internamente. Isto é, a mesma tecnologia que é empregada dentro da instituição, também é utilizada para a comunicação externa, com a Internet.

Para poder compreender como funciona essa comunicação, é preciso que se conheça alguns detalhes dessa arquitetura.

3.1 O Modelo de Camadas TCP/IP

Como o modelo de referência OSI foi construído antes da ligação inter-rede ter sido criada, não contém uma camada para os protocolos inter-redes. Por esse motivo, os mesmos pesquisadores que desenvolveram o TCP/IP, criaram um novo modelo de camadas, o modelo de camadas TCP/IP.

Diferentemente do modelo de referência OSI, que possui 7 camadas, o modelo de camadas TCP/IP, também conhecido como modelo de camadas inter-redes ou modelo de referência de inter-redes, ou ainda modelo de referência TCP/IP [COM 2001], apresenta uma estrutura com 4 camadas, ilustrado na Figura 3.1:

Aplicação: São as aplicações e processos que usam a rede;
Transporte: Oferece um serviço de entrega de dados fim a fim;
Internet: Também conhecida como Inter-rede, define os datagramas e faz o roteamento dos dados;
Interface com a rede: Também chamada de Acesso à Rede, são as rotinas de acesso à rede física.

FIGURA 3.1 - Camadas do Modelo TCP/IP

A camada de aplicação do modelo TCP/IP agrupa as características das camadas aplicação e apresentação do modelo OSI. Nos níveis inferiores, o modelo TCP/IP não especifica padrões de rede a nível físico, mas protocolos para comunicar com padrões existentes [BRA 89] [BRA 89a] [SOC 91]. Porém, de forma geral, o princípio dos modelos é o mesmo.

As aplicações, que residem nos níveis superiores, utilizam a pilha de protocolos para transmitir dados entre si, em entidades diferentes. Os dados a serem transmitidos "descem" até o nível de interface com a rede, adicionando informações de controle, também denominadas cabeçalhos, em cada camada. No momento da chegada dos dados no destino o processo inverso ocorre, onde na "subida", em cada camada são feitas

consistências. Antes de seguir seu curso para a camada superior, o cabeçalho referente a camada atual é retirado.

Para melhor compreensão, a seguir tem-se o detalhamento de cada uma dessas camadas, partindo-se do nível inferior, até o mais alto:

3.1.1 Camada de Interface com a Rede

Esta, que é camada mais baixa da hierarquia do modelo de referência TCP/IP, tem a responsabilidade de garantir a entrega dos dados a outros dispositivos conectados na mesma rede. A interface com a rede é transparente ao usuário.

Quando uma nova tecnologia de *hardware* de rede é criada, novos protocolos da interface com a rede devem ser desenvolvidos, a fim de permitir que as redes que usam TCP/IP possam fazer uso desse novo *hardware*. Desta forma, existem muitos protocolos desta camada, um para cada padrão de rede física [HUN 94].

Neste nível, os datagramas IP são encapsulados em *frames*, para serem transmitidos. O formato não é único, variando de rede para rede, conforme o padrão utilizado. O padrão no qual concentra-se este estudo é o Ethernet, definido pela [HOR 94]. Outra importante função definida nesta camada é o mapeamento dos endereços IP em endereços físicos usados pela rede [PLU 82].

3.1.2 Camada Internet

Acima da camada de interface com a rede está a camada de internet. O principal protocolo associado é o IP (*Internet Protocol*). Este protocolo oferece o serviço básico de entrega de pacotes, o qual a família de protocolos TCP/IP faz uso. Todos os protocolos das camadas superiores utilizam o IP para entregar dados [HUN 94].

As funções do IP incluem:

- Definição do datagrama que é a unidade básica de transmissão;
- Definição do esquema de endereçamento;
- Mover dados entre as camadas de Interface da Rede e Transporte;
- Fazer roteamento de datagramas para *hosts* remotos;
- Fragmentar e unificar datagramas.

O ICMP (*Internet Control Message Protocol*) também faz parte da camada de internet e utiliza o IP para fazer a entrega de mensagens. Seus principais objetivos são comunicar a ocorrência de situações anormais na transferência dos datagramas pela rede e responder a consultas a respeito do estado das máquinas da rede [POS 81a].

O ICMP tem como funções principais:

- Controle de fluxo;
- Detecção de destinos não encontrados;
- Redirecionamento de rotas;
- Verificação de *hosts* remotos conectados.

3.1.3 Camada de Transporte

Os dois principais protocolos da camada de transporte são o TCP (*Transmission Control Protocol*) e o UDP (*User Datagram Protocol*). O primeiro oferece o serviço confiável de entrega de dados, com detecção e correção de erros fim a fim. Já o UDP oferece um serviço de entrega de dados sem conexão. Ambos tem a função de fazer a entrega de dados entre a camada de aplicação e a camada internet.

A opção pela utilização de um ou outro, é feita pelo programador no momento da construção da aplicação, conforme o que julgar mais apropriado.

O detalhamento de cada um dos protocolos é feito mais adiante.

3.1.4 Camada de Aplicação

Esta camada, que é a de nível mais alto dentro da arquitetura do protocolo TCP/IP, inclui todos os processos que usam os protocolos da camada de transporte para entregar os dados.

Novos serviços podem ser acrescentados a qualquer momento a esta camada. Porém os protocolos da camada de aplicação implementados e mais conhecidos são:

- TELNET (*Network Terminal Protocol*): permite login remoto através da rede;
- FTP (*File Transfer Protocol*): utilizado para transferência de arquivos de forma interativa;
- SMTP (*Simple Mail Transfer Protocol*): responsável pela entrega de mensagens eletrônicas;
- DNS (*Domain Name Service*): faz o mapeamento de endereços IP (numérico) para os nomes dos *hosts*.
- RIP (*Routing Information Protocol*): utilizado pelos dispositivos de rede para trocar informações de roteamento;
- NFS (*Network File System*): este protocolo permite o compartilhamento de arquivos por diversos *hosts* na rede.

É importante ressaltar que alguns protocolos citados necessitam da interação do usuário, como o TELNET e o FTP, enquanto outros, como o RIP, podem estar sendo executados mesmo que o usuário não tenha conhecimento algum.

Na Figura 3.2 pode-se identificar as diferentes camadas e os principais protocolos respectivamente implementados. O exemplo ilustrado foi montado sobre o padrão de rede Ethernet, e através desta ilustração é possível identificar como as camadas do TCP/IP ficam acima dos padrões de rede, isto é, da tecnologia de rede utilizada.

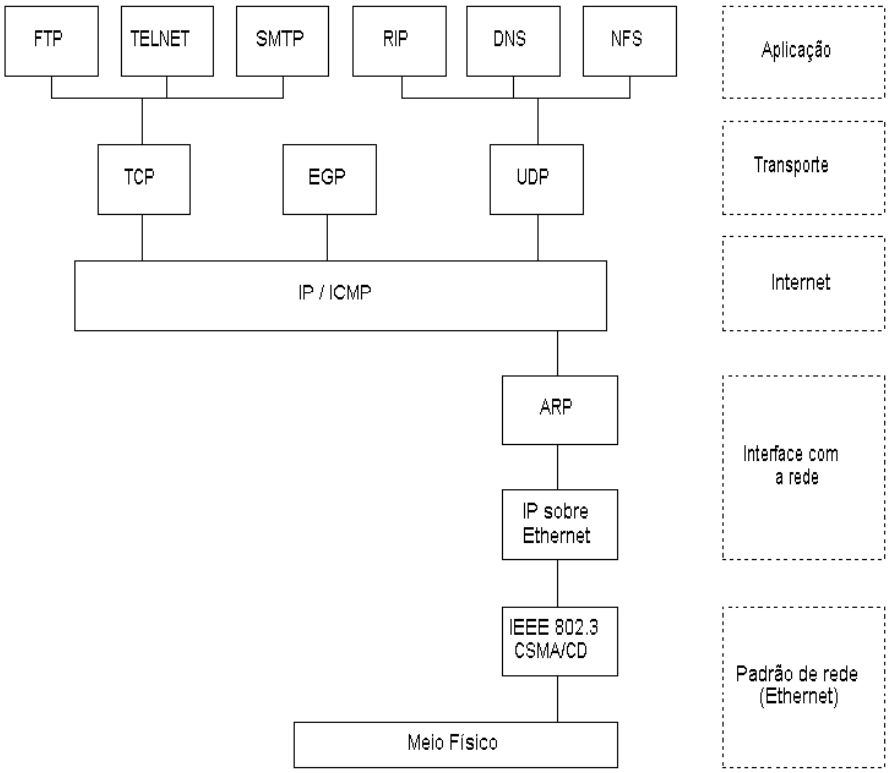

FIGURA 3.2 - Exemplo de protocolos das camadas TCP/IP

Nas seções seguintes são estudados os principais protocolos do conjunto TCP/IP.

3.2 IP – *Internet Protocol*

O IP é um protocolo que realiza a transferência de dados sem a realização de conexões. Em outras palavras, o IP não realiza troca de informações para estabelecer uma conexão fim a fim, antes de iniciar a transferência dos dados. Cada datagrama é independente dos outros. Pacotes enviados de um computador para outro, podem percorrer caminhos diferentes para atingir o mesmo objetivo.

Também é conhecido como protocolo não confiável, porque a entrega não é garantida. Um datagrama pode se atrasar, se perder na rede, chegar duplicado ao destinatário, ou ainda, ser entregue com problemas no conteúdo.O protocolo não tem como verificar quando os dados foram corretamente entregues. Essa função, quando se fizer realmente necessária, fica a cargo de outros protocolos da arquitetura TCP/IP.

Como já foi mencionado, uma das funções do IP é construir os datagramas. Um datagrama é a unidade básica na comunicação TCP/IP. É o formato de pacote adotado.

Um pacote é um bloco de dados que carrega consigo informações necessárias para que ele consiga atingir o destinatário. Analogamente, pode-se compará-lo com uma

carta que é enviada pelo correio convencional. O conteúdo está dentro do envelope, enquanto neste, estão as informações do remetente e do destinatário.

Como cada pacote trafega de forma independente pela rede, o endereço do destinatário é analisado para cada pacote, a fim de definir o caminho que este deve seguir através das redes, para atingir o objetivo final.

As informações de endereçamento são armazenadas no chamado cabeçalho do pacote IP, ilustrado na Figura 3.3. Ele armazena também informações relacionadas ao controle de roteamento e fragmentação, além das que são necessárias para o entendimento do próprio datagrama, como, por exemplo, a versão do IP, o tamanho do cabeçalho (pois este não tem tamanho fixo, o mínimo é 5 palavras de 32 bits) e o tamanho total do datagrama.

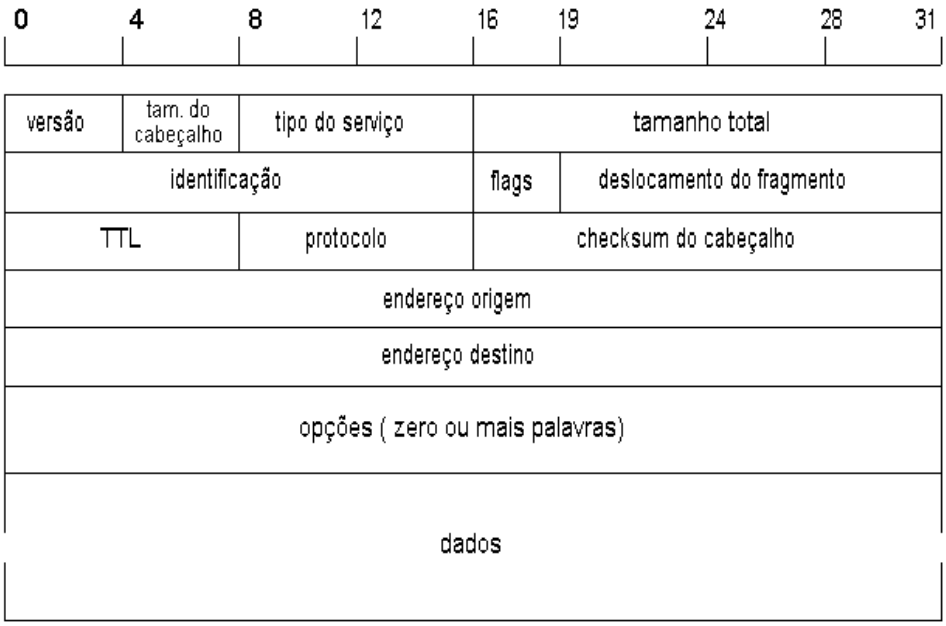

FIGURA 3.3 - Cabeçalho IP

A seguir tem-se o detalhamento das informações contidas no cabeçalho, de forma a compreender a estrutura por completo [POS 81]:

- **Versão:** Indica a versão do protocolo que está sendo utilizado. Neste caso, o protocolo que está sendo descrito é o protocolo versão 4.
- **Tamanho do Cabeçalho:** Informa o tamanho do cabeçalho, que é informado em palavras de 32 bits. O tamanho do cabeçalho pode ser diferente de 5, quando é feito o uso do campo Opções, o que não é muito comum.
- **Tipo de Serviço:** Este campo indica detalhes sobre a qualidade do serviço desejado para a entrega do datagrama IP. A Figura 3.4 representa a estrutura desse campo:

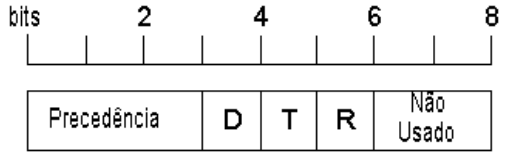

FIGURA 3.4 - Formato do campo Tipo de Serviço

- **Precedência:** Os 3 primeiros bits, denominados precedência, informam a prioridade do datagrama, dependendo do tipo de dado que transporta. Quanto maior o valor, maior a prioridade para a entrega:
  000 – indica um datagrama normal;
  111 – informação de controle.
- **Bits D, T e R**: significam respectivamente *Delay* (atraso), *Throughput* (velocidade) e *Reliability* (confiabilidade). A entidade transmissora pode fazer uso desses bits, ativando-os a fim de exigir baixo atraso, alta velocidade e/ou alta confiabilidade.

  A idéia é passar informações ao roteador, a fim de este possa selecionar diferentes rotas de entrega, conforme a característica do datagrama. Se existir duas rotas entre dois pontos, sendo que uma pode ser lenta e confiável e outra rápida, conforme os bits ativados, o datagrama poderá ser enviado por um ou outro caminho.

  Porém, segundo [TAN 97] e [TOR 2001], o campo tipo de serviço na pratica não é utilizado. Apesar de ser interessante, sua utilização acaba atrapalhando ao invés de ajudar. É difícil diferenciar conceitos como atraso e velocidade. São conceitos muito próximos, enquanto que a confiabilidade também é difícil de medir. Para tal, seria preciso aumentar a complexidade dos roteadores, aumentando seu custo.

  O IETF (*Internet Engineering Task Force*) continua os estudos a fim de melhorar o uso deste campo.

- **Tamanho Total:** Informa o número de bytes que formam o datagrama. Como este campo possui 16 bits, então o tamanho máximo que um datagrama pode assumir é de 65.535 bytes (2^{16}). Por motivos de performance e em função do algoritmo de acesso ao meio físico, isto é, do padrão de rede utilizado que possui regras próprias para o tamanho de quadro, normalmente um datagrama não atinge o valor máximo possível.
- **Identificação:** Identifica cada datagrama. Essa informação é importante para a necessidade de se fazer a fragmentação do datagrama, no caminho até o destino. Todos os fragmentos de um datagrama possuem a mesma identificação.
- **Flags:** Estas informações são utilizadas para controlar a fragmentação do datagrama. Conforme pode-se observar na Figura 3.5 abaixo, é formato por três bits:

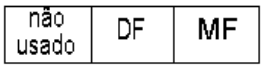

FIGURA 3.5 - Flags do cabeçalho IP

  - O primeiro não é utilizado.
  - O segundo denominado **DF (*Don't Fragment*),** quando ativado, indica que o datagrama não pode ser fragmentado. Redes e equipamentos podem ter diferentes limites para tamanho máximo de datagrama, e com isso a fragmentação poderá ser exigida. Segundo [TAN 97], todas as máquinas devem aceitar fragmentos de até 576 bytes.
  - O bit **MF (*More Fragments*)** significa mais fragmentos. É utilizado para indicar qual é o último fragmento. Todos os fragmentos, exceto o último, possuem esse bit ativado (igual a 1), indicando que existem mais fragmentos posteriores.
- **Deslocamento do Fragmento (*Fragment Offset*):** também utilizado para o controle da fragmentação, esse campo informa a ordem dos fragmentos. Este campo é utilizado para situar o fragmento dentro do datagrama. Ele conta os fragmentos em blocos de 8 bytes. Imaginando uma seqüência de fragmentos de 600 bytes, o primeiro tem *offset* igual a zero, o segundo um *offset* igual a 75 (600 / 8 = 75). Por esta razão todos os fragmentos, com exceção do último, devem ter um tamanho múltiplo de 8.
- **TTL (Time To Live):** Indica o tempo de vida máximo para o datagrama, em segundos. O valor máximo é igual a 255. Quando o valor chegar a zero, o datagrama deve ser descartado. Este campo é decrementado a cada vez que o cabeçalho é processado. A cada *gateway* que passa, o campo é decrementado de uma unidade, mesmo que esta passagem leve apenas alguns milisegundos A idéia é evitar a ocorrência de datagramas "sem rumo", que por problemas de rotas, fiquem navegando de forma infinita pela rede, congestionando-a.
- **Protocolo:** Com a informação aqui armazenada, pode-se identificar o protocolo que fez, ao IP, a solicitação de envio do datagrama. Podendo assumir valores até 255, os principais protocolos aqui referenciados são [REY 94]:
  - 1 – ICMP
  - 2 - IGMP
  - 6 – TCP
  - 17 – UDP
- ***Checksum* do Cabeçalho:** É um verificador do cabeçalho. A cada salto do pacote, esse campo é analisado e recalculado, uma vez que no mínimo o campo TTL é alterado.
- **Endereço Origem:** Especifica o endereço IP do equipamento gerador do datagrama.
- **Endereço Destino:** Informa o endereço IP da entidade para qual o datagrama está endereçado.
- **Opções:** Este campo é opcional. É utilizado para teste e verificação de erros em uma rede. As funções mais importantes são traçar a rota de rede que está sendo usada, entre e origem e o destino, e marcar o horário (com precisão de milisegundos) em que o datagrama passa por cada roteador. A utilização

deste campo também não é muito comum, ficando desta forma, o cabeçalho com tamanho de 5 palavras de 32 bits. Se esse campo possuir tamanho menor que 32 bits ou seus múltiplos, então são adicionados zeros (chamados *pad*) até que atinja o tamanho desejado. Em [POS 81] pode-se obter um detalhamento maior sobre as funções e o uso deste campo.

- **Dados:** Este não faz parte do cabeçalho propriamente dito. É a área "útil" do datagrama. Aqui ficam os dados que estão sendo transportados. O verdadeiro propósito da existência do datagrama IP. São estas as informações que são repassadas para o protocolo indicado no campo chamado "protocolo".

3.2.1 Roteamento dos Datagramas

O roteamento dos datagramas através da rede é feito pelo IP, em função dos endereços de rede das máquinas origem e destino, especificado no cabeçalho do próprio datagrama. Estes endereços, conhecidos como endereços IP identificam os computadores de forma única na rede.

Quem tem o papel de definir a rota pela qual um datagrama deve seguir, a fim de chegar ao destino final é o roteador IP. Estes equipamentos que são utilizados para interligar redes, podem ser formados por um *hardware* especificamente criado para tal, ou um computador configurador com esta finalidade.

Podem ocorrer problemas na escolha da rota pela qual o datagrama deverá navegar, e com isso este se "perder" na rede. Tais problemas podem ser ocasionados por equipamentos mal configurados.

Para que estes datagramas não fiquem navegando "eternamente", o IP oferece um campo chamado tempo de vida (TTL), que é utilizado para fazer o controle de saltos que o datagrama faz. Quando o datagrama é gerado, é atribuído a este campo, um valor que no máximo deve ser 255. A cada roteador, este campo é decrementado, sendo o datagrama descartado quando o TTL chegar a zero.

Quando o datagrama chega ao destino, as informações contidas no seu campo de dados são passadas para um protocolo em uma camada acima. Como o IP pode transportar informações para protocolos diferentes, faz-se uso do campo protocolo do cabeçalho IP para identificar o protocolo que originou os dados.

3.2.2 Tamanho do Datagrama e Fragmentação

Durante a trajetória de um datagrama, ele poderá passar por uma rede em que o tamanho máximo aceito para um datagrama (MTU – *Maximum Transmission Unit*) é inferior ao seu próprio tamanho (a Tabela 3.1 apresenta alguns tipos de rede e os respectivos MTUs.). Isto é, o datagrama possui um tamanho maior do que o máximo permitido para a rede em que está trafegando.

TABELA 3.1 - MTUs

Rede	MTU (em octetos)
Ethernet	1.500
Token Ring	4.096
FDDI	4.470

Nessas situações, o datagrama é dividido em fragmentos para que possa passar pela rede, a isso chama-se fragmentação.

A fragmentação consiste em criar novos datagramas menores, sendo que a área de dados do datagrama original é dividida entre eles. Um fragmento, ao passar em uma rede com MTU menor pode ser fragmentado novamente, e mesmo assim o receptor conseguirá remontar o datagrama original. Na fragmentação, a maioria dos dados do cabeçalho é copiada para os cabeçalhos dos novos datagramas. O cabeçalho abriga três informações importantes para o procedimento de recuperação de um datagrama fragmentado:

- Identificação do datagrama;
- Deslocamento do fragmento;
- Indicação de último.

Todos os fragmentos possuem o mesmo identificador de datagrama, copiado do cabeçalho do datagrama original.

Através do deslocamento do fragmento, pode-se identificar a posição relativa do fragmento no datagrama original. Isso é importante porque muitas como cada fragmento pode seguir uma rota diferente, podem chegar fora de ordem, necessitando de ordenação no momento da remontagem.

A indicação de último fragmento é feita por meio do *flag* MF (*more fragments*), que é ativado (igual a 1) em todos os fragmentos, com exceção do último.

Como desvantagem, a fragmentação exige que o receptor tenha memória suficiente para armazenar todos os fragmentos conforma chegam, antes de remontar o datagrama, para poder retirar os dados e repassar ao protocolo da camada superior. O problema não se dá em relação ao tamanho ou quantidades de fragmentos de um único datagrama, mas em determinado momento, o receptor poderá estar aguardando fragmentos de diversos datagramas.

Caso um fragmento seja perdido na rede, o datagrama não será entregue. Com isso, o protocolo acima do IP solicitará a retransmissão, que será feita de forma integral do datagrama e não somente do fragmento perdido.

3.3 ICMP - *Internet Control Message Protocol*

Um pacote ICMP, *Internet Control Message Protocol*, é um pacote da camada de internet do modelo TCP/IP, cujo propósito, não é trocar informações entre programas, mas sim, entre os computadores origem e destino de uma comunicação, ou ainda, destes com os equipamentos intermediários na comunicação [COM 98].

Como já foi estudado, o IP é um protocolo não confiável. Sendo assim, o ICMP tem o propósito de fornecer informações sobre possíveis problemas na comunicação. Desta forma, não se preocupa em corrigir os erros e também não tem a função de verificar a integridade dos datagramas que circulam pela rede.

Por exemplo, quando um roteador fica impossibilitado de enviar adiante um datagrama recebido, por estar congestionado demais ou simplesmente por ter sido zerado o campo TTL do datagrama, ele faz uso do ICMP para informar ao transmissor da ocorrência do problema. Dito isso, pode-se concluir que o ICMP, mesmo que estudado em separado, é uma parte integrante do protocolo IP.

Como pode ser observado na Figura 3.6, o pacote ICMP trafega sobre o IP, embutido no campo de dados do mesmo. Uma vez visto isso, pode-se afirmar que mesmo em relação a um pacote ICMP, que busca avisar o transmissor sobre problemas com pacotes enviados, não existe nenhuma garantia de que ele chegue corretamente ao seu destino, podendo ser perdido no meio do caminho.

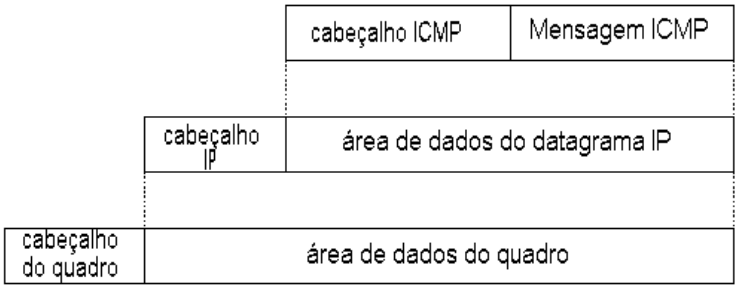

FIGURA 3.6 - Encapsulamento de uma mensagem ICMP

O formato das mensagens ICMP varia. O primeiro campo é chamado tipo de mensagem. O segundo, denominado código, está vinculado ao primeiro. Para cada tipo de mensagem, existem vários códigos que representam situações diferentes. Isto é, para cada tipo de mensagem pode-se ter diferentes valores para o código. Com a combinação dos dois campos, tem-se a definição de diferentes mensagens.

Conforme especificado em [POS 81a], [MOG 85], [BRA 89] e [COM 98], a Tabela 3.2 apresenta os tipos de mensagem em uso:

TABELA 3.2 - Tipos de mensagens ICMP

Tipo	**Descrição**
0	Resposta à Mensagem de Eco (*Echo Reply*);
3	Aviso de Destino Inacessível;
4	Solicitação de Redução de Índice de Transmissão;
5	Solicitação de Redirecionamento;
8	Mensagem de Eco (*Echo Request*);
11	Tempo de Vida de um Datagrama Excedido;
12	Problema de Parâmetros de um Datagrama;
13	Solicitação de Indicação de Hora;
14	Resposta à Indicação de Hora;
15	Solicitação de Informação (obsoleto)
16	Resposta de Informação (obsoleto)
17	Solicitação de Máscara de Endereçamento;
18	Resposta à Solicitação de Máscara de Endereçamento.

Cada o tipo de mensagem, por tratar de assunto específico, possui a necessidade de transportar diferentes informações relacionadas. Por esta razão, cada tipo de mensagem, gera um formato diferente de mensagem ICMP.

Quando a mensagem está informando um erro ocorrido com o datagrama (tipo 4, 5 ou 12), carrega consigo o cabeçalho e os primeiros 64 bits da área de dados do datagrama problemático, a fim de proporcionar maior precisão para o receptor da mensagem de erro.

Para melhor compreensão do ICMP, cada tipo de mensagem é aprofundado a seguir.

3.3.1 Mensagens de Eco

A Figura 3.7 mostra o formato das mensagens de solicitação e de resposta de eco.

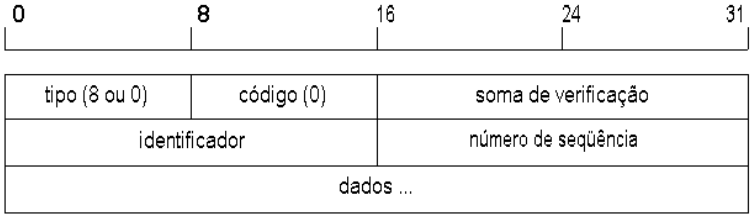

FIGURA 3.7 - Mensagem ICMP de eco

Estas mensagens são utilizadas para verificar a existência de um caminho entre o transmissor e o receptor. O transmissor envia um datagrama contendo uma mensagem ICMP de requisição de eco, e, no momento que o destinatário a recebe, gera outra mensagem ICMP de resposta, incluindo dados recebidos no datagrama original. Este é o tipo de mensagem gerada pelo comando ping.

Os campos identificador e número de seqüência são utilizados para associar as respostas com as requisições feitas. O identificador é utilizado como um identificador de sessão, já o número de seqüência é incrementado a cada nova requisição feita. As respostas são montadas com os valores originais da mensagem de requisição.

A soma de verificação está presente em todos os tipos de mensagem ICMP.

3.3.2 Destino Inacessível

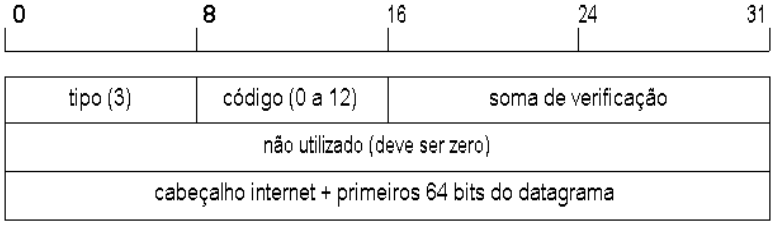

FIGURA 3.8 - Mensagem ICMP de destino inacessível

Caso um roteador não consiga entregar um determinado datagrama, uma mensagem ICMP tipo 3, que possui a estrutura ilustrada na Figura 3.8, é enviada de volta para o transmissor. Mesmo que o IP seja um protocolo sem garantia de entrega, existe a preocupação com os datagramas descartados. É importante que o transmissor seja informado, com maiores detalhes, sobre qual o destino que está inacessível, e o respectivo motivo.

Os motivos são descritos e detalhados através do campo código que pode assumir valores de 0 a 12 conforme listado abaixo:

0 – Rede inacessível;
1 – Máquina inacessível;
2 – Protocolo inacessível;
3 – Porta inacessível;
4 – A rede exige fragmentação, mas o DF está ativado;
5 – Falha na rota de origem;
6 – Rede de destino desconhecida;
7 – Máquina destino desconhecida;
8 – Máquina de origem isolada;
9 – Comunicação com a rede destino está proibida pela administração da rede;
10 – Comunicação com a máquina destino está proibida pela administração da rede;
11 – Rede inacessível para o tipo de serviço solicitado;
12 – Máquina inacessível para o tipo de serviço solicitado.

Com este grupo de códigos é possível identificar o motivo pelo qual o datagrama não conseguiu atingir o seu destino. A informação que indica qual é a máquina que não pode ser acessada, é obtida analisando o cabeçalho do datagrama enviado, que retorna no corpo do datagrama ICMP.

A maioria destas mensagens é enviada por roteadores existentes no caminho, mas algumas, como os códigos 2 e 3, são enviadas por *hosts.*

As falhas na entrega detectadas pelos roteadores são informadas ao emissor, porém, em certas ocasiões eles podem não detectar que houve um problema de entrega. Um exemplo claro disso, ocorre quando a máquina destino, que está em uma rede Ethernet recebendo os pacotes normalmente, é desligada. O hardware de rede não tem mecanismo para avisar o roteador do desligamento de um computador. Com isso, o roteador pode continuar enviando pacotes destinados a um computador que já não está em operação, não detectando os problemas de entrega.

Mesmo com esse problema, este tipo de mensagem pode informar com sucesso grande parte dos problemas de entrega de datagramas.

3.3.3 Solicitação de Redução de Índice de Transmissão

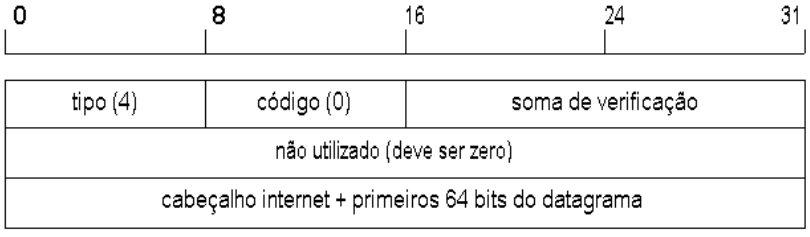

FIGURA 3.9 - Mensagem ICMP de redução de índice de transmissão

Utilizado para controle de congestionamento.

Quando um roteador ou um *host* recebe um número maior de datagramas do que ele tem capacidade de processar, caracteriza-se um congestionamento.

Uma vez que os datagramas chegam com muita rapidez para serem processados por um host ou por um roteador, estes são enfileirados temporariamente na memória. Se essa for uma situação temporária, os enfileiramentos resolvem o problema. Mas quando a situação é contínua, a memória é esgotada e os datagramas que chegam passam a ser descartados.

Nessa situação, para cada datagrama descartado, uma mensagem ICMP (*source quench*) é enviada ao transmissor para que este reduza a sua velocidade de transmissão. No cabeçalho dessa mensagem, representado pela Figura 3.9, segue também o cabeçalho e mais os 64 primeiros bits do datagrama descartado, a fim do emissor poder fazer o reconhecimento completo da comunicação que está congestionada.

É importante observar que não existe mensagem de aviso para que o emissor volte a transmitir na velocidade original, quando o congestionamento está controlado. O que ocorre, é que o transmissor reduz o índice de remessa de datagramas até que deixe de receber mensagens *source quench*. A partir desse momento, o índice cresce enquanto não receber outra mensagem desse tipo.

3.3.4 Solicitação de Redirecionamento

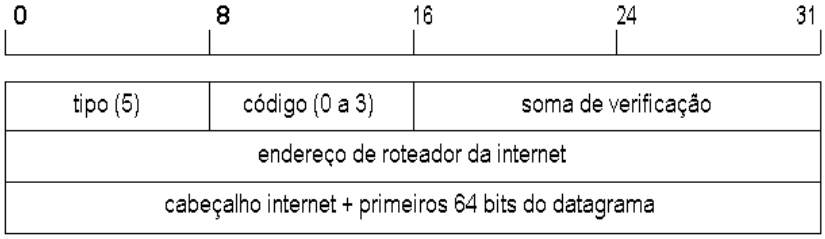

FIGURA 3.10 - Mensagem ICMP de redirecionamento

Ao receber um datagrama da rede local a qual está conectado, o roteador pode identificar uma rota melhor a ser usada para que este atinja o destino. Nesse caso, envia uma mensagem de solicitação de redirecionamento ao transmissor, envia também o datagrama ao destino. Essa categoria de mensagem ICMP é utilizada somente dentro da rede local.

A grande vantagem do esquema de redirecionamento do ICMP é a simplicidade. Permite que as máquinas inicializem, conhecendo apenas o endereço do roteador principal. Este, durante a comunicação, passa a informar ao *host* as melhores rotas na rede onde está conectado.

As variações das mensagens de redirecionamento são:

0 - Redirecionar datagrama para a determinada rede;
1 - Redirecionar datagrama para determinado *host (mais usada);*
2 - Redirecionar datagrama para o tipo de serviço e rede;
3 - Redirecionar datagrama para o tipo de serviço e *host.*

No cabeçalho da mensagem ICMP, Figura 3.10, segue somente o endereço do roteador indicado como melhor rota, acompanhado do cabeçalho e dos primeiros 64 bits de dados do datagrama IP gerador do aviso.

3.3.5 Tempo de Vida de um Datagrama Excedido

A Figura 3.11 ilustra o formato de uma mensagem ICMP utilizada para avisar o emissor sobre descartes de pacotes.

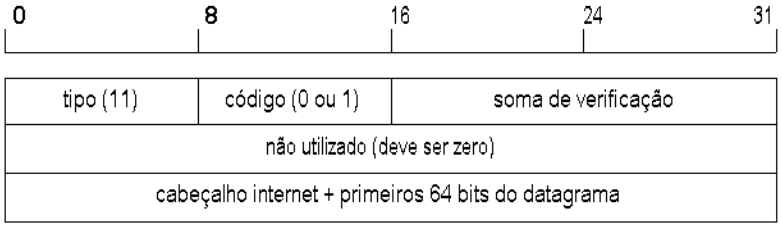

FIGURA 3.11 - Mensagem ICMP de tempo de vida do datagrama excedido

Como visto anteriormente, o datagrama IP possui um campo denominado tempo de vida, que é decrementado a cada vez que passa por um roteador. Quando este contador for zerado, o pacote é descartado e é enviada para a máquina transmissora, uma mensagem ICMP indicando que o tempo de vida foi excedido.

Outra situação em que o datagrama é descartado ocorre quando este fica excessivamente nas filas de espera para remontagem de pacotes fragmentados. Assim que o primeiro fragmento de um datagrama chegar ao destino, o *host* dispara um temporizador. Se este expirar antes que o pacote seja totalmente remontado, então será descartado, gerando uma mensagem ICMP de aviso ao emissor.

Existe um código específico para identificar cada uma das situações acima:

0 - Número excessivo de saltos;

1 - Tempo de remontagem dos fragmentos excedido.

O primeiro indica que o datagrama estava “perdido” na rede, tendo passado por um número excessivo de roteadores, sem atingir o destino.

O segundo se dá quando ocorre um problema para coletar todos os fragmentos de um datagrama.

3.3.6 Problemas de Parâmetros de um Datagrama

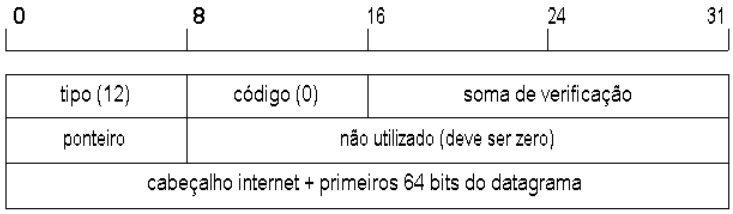

FIGURA 3.12 - Mensagem ICMP de problemas de parâmetros de um datagrama

Os tipos de mensagens ICMP vistas até aqui procuram cobrir os problemas mais comuns que ocorrem com datagramas. Porém podem existir situações em que o *host* ou o roteador encontram problemas que não se encaixam em nenhuma das situações previstas.

Esse tipo de erro muitas vezes se dá pela ocorrência de argumentos incorretos no cabeçalho do datagrama. Para estes casos, existe este tipo de mensagem.

Para facilitar a identificação do problema, no cabeçalho destas mensagens segue um campo chamado ponteiro, conforme pode ser observado na Figura 3.12 acima. O transmissor utiliza este campo para indicar o octeto do datagrama original, onde ocorreu o erro. A contagem de octetos inicia em zero, isto é, quando o apontador indicar o octeto 1, estará indicando problemas no campo tipo de serviço.

3.3.7 Solicitação de Indicação de Hora

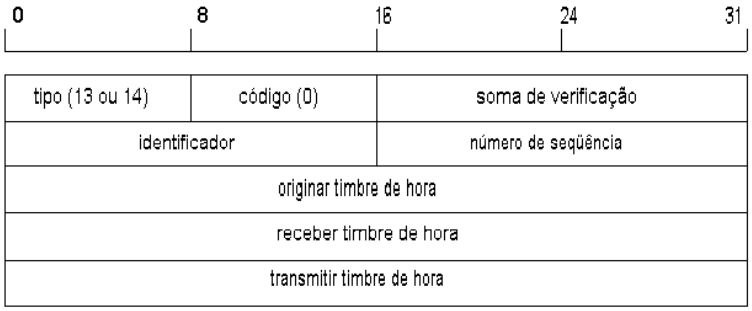

FIGURA 3.13 - Mensagem ICMP de indicação de hora

Mesmo estando em uma rede, as máquinas geralmente operam de forma independente, tendo a possibilidade de trabalharem com noções próprias de tempo. Dependendo da aplicação, é importante manter os relógios sincronizados entre os diferentes computadores. Através de mensagens de controle, uma máquina pode pedir o horário do relógio de outra.

A máquina solicitante dispara uma mensagem de solicitação de indicação de hora ICMP (tipo 13) a outra máquina. A segunda retorna uma mensagem de resposta de solicitação de hora (tipo 14).

No cabeçalho desse tipo de mensagem, ilustrado na Figura 3.13, existe dois campos denominados identificador e número de seqüência, que são utilizados para associar as mensagens de resposta de horário com as solicitações previamente feitas. O primeiro é utilizado como um identificador de sessão, já o número de seqüência é incrementado a cada nova requisição feita. As respostas são montadas com os valores originais da mensagem de requisição.

Além destes, existem mais três campos importantes para o funcionamento desse mecanismo de solicitação de hora: o campo originar timbre de hora é preenchido pelo solicitante imediatamente antes de transmitir o pacote; o campo receber timbre de hora que é preenchido pelo receptor da solicitação, imediatamente após o seu recebimento; e finalmente, o campo transmitir timbre de hora que é preenchido imediatamente antes da transmissão da resposta. Estes três campos são utilizados para calcular as estimativas de tempo.

O uso dessas mensagens serve para sincronizar o relógio de duas máquinas ou ainda para medir o tempo de resposta da rede, isto é, o tempo que um datagrama leva para atingir um determinado destino.

Como a estimativa exata de duração de um retorno de transmissão é difícil, em função das variações nos tempos de transmissão entre duas máquinas, a utilização de

mensagens ICMP para indicação de hora é restrita. É impossível garantir uma sincronia 100% perfeita entre ambas.

3.3.8 Solicitação e Resposta de Informação

Este tipo de mensagem (tipo 15 e 16), são consideradas obsoletas. Foram inicialmente utilizadas para que um computador obtivesse seu endereço dentro da rede no momento da inicialização do sistema.

Hoje estas funções são executadas por outros protocolos, como RARP e BOOTP.

3.3.9 Obtenção de Máscara de Endereçamento

A Figura 3.14 a seguir, representa o formato de uma mensagem de solicitação ou resposta de máscara de endereçamento.

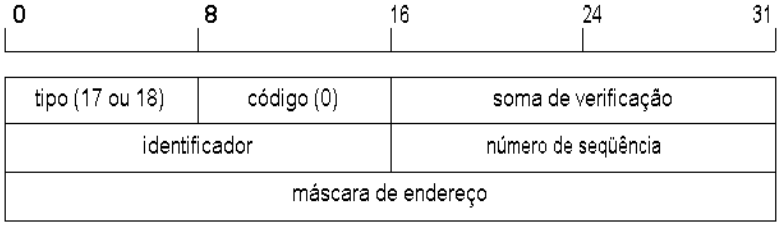

FIGURA 3.14 - Mensagem ICMP de obtenção de máscara de endereçamento

Estas mensagens são utilizadas pelo *host* para solicitar à rede a máscara de endereçamento que deve ser utilizada. A mensagem de solicitação (tipo 17) pode ser enviada diretamente ao roteador principal, se a máquina solicitante souber o endereço do mesmo. Caso contrário, a mensagem é difundida para toda a rede.

Esse tipo de mensagem também possui no cabeçalho os campos identificador e número de seqüência, a fim de permitir que as respostas sejam associadas às solicitações previamente feitas. Possui também um campo específico para armazenar os 32 bits referentes a máscara solicitada.

Com o detalhamento de cada tipo de mensagem ICMP pode-se observar que cada um desses tipos possui uma aplicação específica. Através do código que identifica o tipo de mensagem ICMP, pode-se reconhecer se esta é:

- um comando;
- uma resposta a um comando;
- uma informação de *status*;
- uma condição de erro.

Deve-se observar também que, ao contrário do UDP e do TCP, as mensagens ICMP não contém informações de portas de origem e destino.

3.4 TCP - *Transmission Control Protocol*

Segundo [ALB 2001], o *Transmission Control Protocol* (TCP) é o mais importante e complexo protocolo de transporte da família TCP/IP. É utilizado pela grande maioria dos serviços de rede.

O TCP usa o serviço de datagramas não-confiável oferecido pelo IP para enviar dados a outro computador, mas oferece um serviço confiável de entrega de dados para os programas aplicativos.

O estudo do cabeçalho TCP, que está ilustrado na Figura 3.15, é importante para a compreensão desse protocolo [POS 81b]. Cada campo é destacado a seguir:

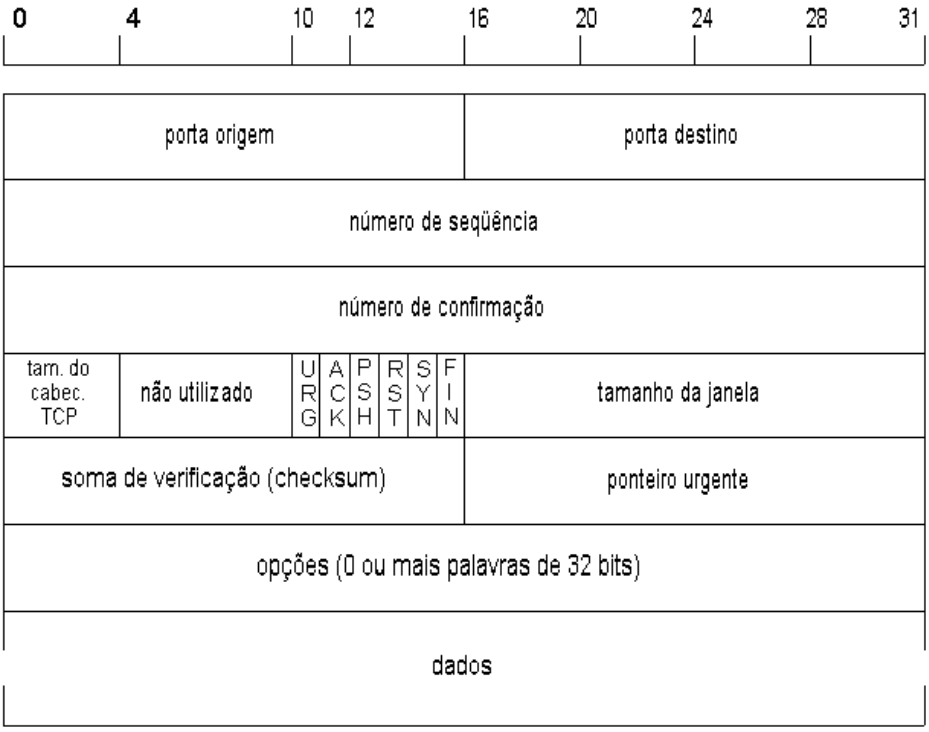

FIGURA 3.15 - Cabeçalho TCP

- **Porta Origem e Porta Destino**: Estes, que são os primeiros campos do cabeçalho TCP indicam o número das portas que estão sendo utilizadas pela conexão. O primeiro na máquina que originou o datagrama e segundo para a porta a qual ele é destinado. A sessão 3.6 a seguir traz mais detalhes sobre portas de serviço.
- **Número de Seqüência** (*Sequence Number*): Quando o *flag* SYN, que será detalhado a seguir, não estiver presente, este campo armazena o número de seqüência do primeiro octeto de dados do datagrama atual dentro do fluxo total de dados gerado pela aplicação. Exemplificando: se o último datagrama recebido da aplicação tinha um número de seqüência

1200, e a área de dados do mesmo era de 250 bytes, então o atual deve ter um número de seqüência igual a 1450. Caso o SYN esteja presente, o campo número de seqüência armazena o número inicial aleatório (X) para a seqüência que está iniciando e o primeiro byte de dados será numerado em (X+1).

- **Número de Confirmação** (A*cknowledgement Number*): É utilizado com o *flag* ACK. Nesse campo é colocado o número de seqüência do próximo segmento que o receptor espera receber. Tomando o mesmo exemplo utilizado no parágrafo anterior, após receber um datagrama com número seqüência 1200 e área de dados de 250, o *host* gerará um datagrama de confirmação recebimento, a ser enviado ao transmissor, com o campo número de confirmação igual 1450, indicando que recebeu o datagrama anterior e está aguardando o datagrama com número de seqüência 1450.
- **Tamanho do Cabeçalho** (*Header Length*): especifica quantas palavras de 32 bits compõem o cabeçalho TCP. Isso é necessário porque o campo *options*, que faz parte do cabeçalho, possui um tamanho variável. Esse dado é importante para definir onde começa a área de dados dentro do datagrama.
- **Área reservada**: É uma área de seis bits que não é utilizada.
- **Bits de controle** (*Flags*): São seis bits usados para controle, conforme especificado a seguir, segundo [TAN 97]:
    - URG: Este bit é ativado quando o campo ponteiro urgente possuir um valor válido que deve ser utilizado.
    - ACK: Quando ativado, indica que é uma confirmação e o campo número de confirmação possui um valor válido.
    - PSH: Indica dados com *flag PUSH*. Quando os datagramas são recebidos, os dados são extraídos e armazenados em um *buffer*, sendo repassados para as camadas superiores (aplicação) somente quando este estiver preenchido. Porém, quando o receptor identificar que o datagrama está com o *PSH* ativado, entregará os dados a aplicação imediatamente, não aguardando o preenchimento completo do *buffer*.
    - RST: Serve para reiniciar uma conexão em andamento, ou também para negar uma solicitação de conexão.
    - SYN: Significa sincronismo. É utilizado no momento do estabelecimento de conexões para indicar que o número de seqüência deve ser sincronizado, com o valor contido no campo respectivo.
    - FIN: Utilizado para encerrar uma conexão. Indica que o transmissor não tem mais dados a enviar.
- **Tamanho da Janela**: Todo o fluxo TCP é controlado fazendo uso de uma janela deslizante de tamanho variável. Com este campo, o receptor indica ao transmissor quantos bytes podem ser enviados a partir do byte confirmado. Se este campo possuir valor igual a zero, indica que todos os dados até o que indicado por (número de confirmação –1) foram recebidos, porém, o receptor solicita que o transmissor pare de enviar dados momentaneamente. A permissão para a retomada da transmissão pode ser feita posteriormente com o envio de um datagrama com o

mesmo número de confirmação e com o campo tamanho da janela maior que zero.

- **Soma de Verificação** (***Checksum***): Este campo contém uma soma de verificação, a fim de aumentar a confiabilidade do TCP.
- **Ponteiro Urgente**: Utilizado em conjunto com o *flag* URG, seu uso é feito quando existem dados que precisam ser processados com maior urgência. Ele indica a posição dentro do segmento onde os dados urgentes terminam. O transmissor utiliza esse recurso para especificar dados urgentes, de modo que o programa receptor deva ser notificado de sua chegada o mais rápido possível, independente de sua posição no fluxo de dados. Após todos os dados urgentes consumados, o TCP diz ao programa aplicativo para voltar à operação normal.
- **Opções**: Esse campo é opcional e possui tamanho variável. Se esse campo possuir tamanho menor que 32 bits ou seus múltiplos, então são adicionados zeros (chamados *pad*) até que atinja o tamanho desejado. Esse campo é pouco utilizado, e normalmente é usado para que o transmissor e o receptor troquem informações sobre o tamanho máximo do segmento que será utilizado na conexão.

Como já mencionado, este é o protocolo mais importante da família TCP/IP. Este é o protocolo responsável por identificar e corrigir perdas de segmentos, segmentos fora de ordem e segmentos com informações incorretas, isto é, oferece funções como controle de fluxo, controle de erros e troca de informações de *status*. O TCP é o responsável pela solicitação de retransmissão que poderá ser necessária em caso de perda de pacotes [HEL 99].

A seguir serão detalhados os principais procedimentos operacionais do TCP.

3.4.1 Abertura de Conexões

Conexão é a denominação dada para a comunicação entre duas aplicações em duas máquinas diferentes. Isto é, uma conexão fica associada a dois endereços IP (que representam os hosts *envolvidos*) e para cada um deles, uma porta de serviço que terá função cliente ou servidora.

O processo de abertura de conexão pelo TCP é conhecido como *handshake* (aperto de mão) de três vias. É nesse processo que as duas entidades envolvidas irão confirmar o número de seqüência inicial a ser utilizado na conexão.

Para melhor compreensão do processo, o *host* que inicia a abertura da conexão passa a ser denominado transmissor, enquanto o outro, passa a ser denominado receptor.

O transmissor envia um pacote contendo apenas o número de seqüência (X) definido aleatoriamente e o SYN ativado. O receptor pega o número de seqüência e confirma o recebimento do pacote, respondendo com um pacote contendo o seu número de seqüência aleatório Y, o campo numero de confirmação com o valor (X+1), e os *flags* SYN e ACK ativados. No momento que o transmissor receber este pacote, envia um pacote de confirmação de recebimento de volta ao receptor, com o número de confirmação igual a (Y+1) e o somente o *flag* ACK ativado.

A Figura 3.16 ilustra a seqüência de troca de mensagens necessárias para efetuar o estabelecimento da conexão. No exemplo, o transmissor utiliza o número de seqüência inicial igual a 100 e o receptor utiliza 300.

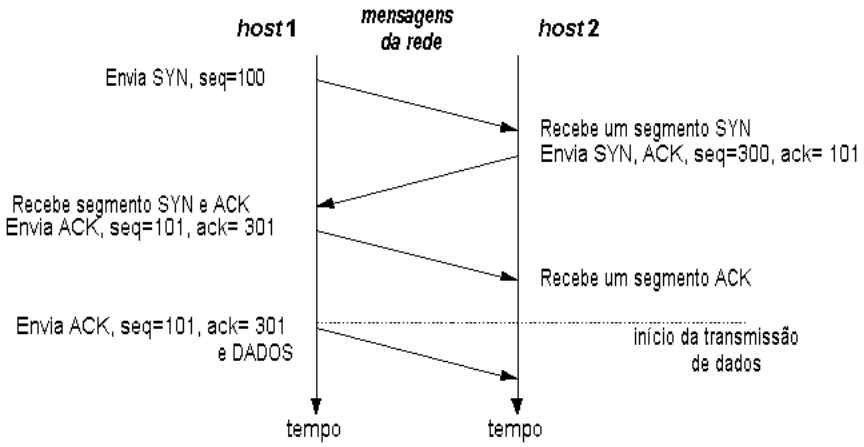

FIGURA 3.16 - Estabelecendo uma conexão

Feito isso, inicia-se o processo de transmissão de dados propriamente dito, que pode ser observado na última mensagem da Figura 3.16.

3.4.2 Encerramento de Conexões

Se nenhum problema ocorrer, ao final, quando não existirem mais dados a serem transmitidos, o transmissor solicitará o encerramento da conexão. Esse processo também é feito usando-se um “aperto de mão”.

O encerramento é feito em cada um dos dois sentidos da transmissão. Quando uma conexão tiver sido concluída em determinada direção, o TCP não aceitará mais dados naquele sentido. Porém, no sentido oposto, os dados poderão continuar fluindo por algum tempo.

A Figura 3.17 representa o procedimento para encerramento de conexão. O *host* A envia um pacote com o *flag* FIN ativado (mensagem 1). Quando o *host* B processar o pacote (na devida ordem de seqüência e não na ordem de chegada), informa a aplicação que os dados chegaram ao fim e envia um pacote de confirmação ao transmissor, mas com o *flag* FIN desativado (mensagem2). Quando a aplicação instrui o TCP a encerrar a conexão, o *host* B envia um pacote com o FIN ativado para o host A (mensagem 3), que confirma o recebimento do mesmo (mensagem 4). A partir deste momento, as conexões em ambos os sentidos estarão encerradas.

Pode ainda ocorrer uma variação deste procedimento. As mensagens 2 e 3 podem ser unidas em uma só. Isto é, ao mesmo tempo em que o *host* B confirma, com ACK, o recebimento do FIN, já envia o seu FIN para o *host* A.

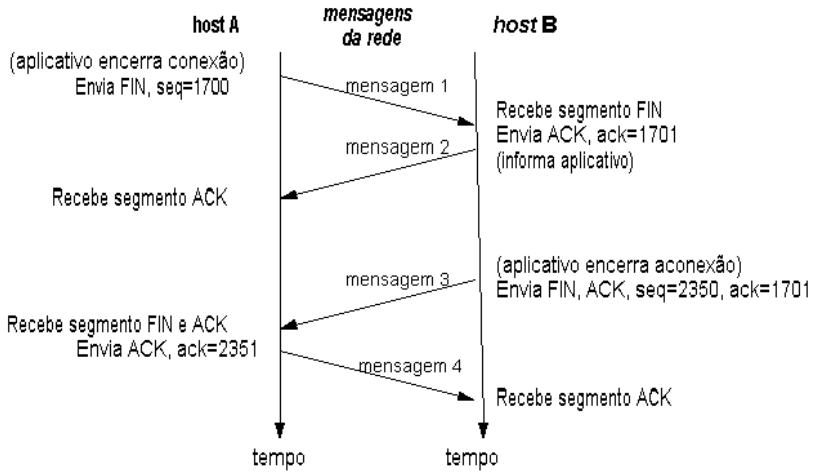

FIGURA 3.17 - Encerrando uma conexão

O procedimento visto acima, com o uso do FIN, é o método normal de encerramento de uma conexão. Porém podem surgir condições anormais que forçam um programa aplicativo ou um software de rede a interromper uma conexão [COM 98]. Para essa operação usa-se o *flag* RST.

Para iniciar a interrupção de uma conexão, basta que qualquer uma das partes envie um segmento com RST ativado. A outra extremidade imediatamente aborta a conexão. Com o *reset*, ocorre um abortamento instantâneo, a transferência em ambas as direções cessa imediatamente e os *buffers* são liberados.

3.4.3 Transmissão e Recepção de Pacotes

Como já foi visto, o TCP oferece um serviço confiável de entrega de dados, sem duplicação ou perdas. Para isso, usa uma técnica conhecida como confirmação positiva com retransmissão. Nessa técnica, assim que o receptor receber um pacote, ele deve enviar ao transmissor uma mensagem de confirmação de recebimento. Já o transmissor, quando envia um pacote, ativa um temporizador. Se este se esgotar antes de receber a confirmação por parte do receptor, então o pacote é retransmitido.

Para manter um controle sobre os dados enviados, buscando uma garantia de que eles chegaram ao seu destino, o TCP faz uso do *flag* ACK.

Na mensagem de confirmação, o ACK está ativado e o campo número de confirmação especifica o número de seqüência do próximo datagrama aguardado pelo receptor.

Se em um determinado tempo tal confirmação não ocorrer, então o transmissor deduz que o datagrama IP que carregava as respectivas informações inseridas pelo TCP foi descartado no caminho e não chegou ao destino. Com base nessa dedução o pacote é retransmitido.

O tempo de espera pela confirmação é definido dinamicamente pelo protocolo TCP, pois em redes grandes não é viável se utilizar tempos fixos [TOR 2001]. O TCP faz a medição do tempo decorrido entre a transmissão de um datagrama e o recebimento

de sua confirmação. Na verdade o tempo de espera é definido em relação a média dessas cronometragens.

A definição do tempo de espera é algo crítico. Se for alto, então perde-se muito tempo para identificar uma perda de dados e o processo de comunicação atrasa. Caso o tempo seja reduzido, surge o risco de se retransmitir um datagrama que ainda está preso no tráfego. Nesse último caso, tem-se a ocorrência de pacotes duplicados no receptor, que também deve ser tratado no nível do TCP.

O controle dos dados transmitidos e recebidos não é feito através de uma numeração seqüencial para os pacotes, mas sim, cada pacote possui um número que identifica o primeiro byte de dados do pacote dentro do fluxo de dados que está sendo gerado naquela conexão TCP. Tal numeração é armazenada no campo número de seqüência.

O campo número de confirmação carrega a informação do número de seqüência do próximo datagrama esperado, dentro da conexão.

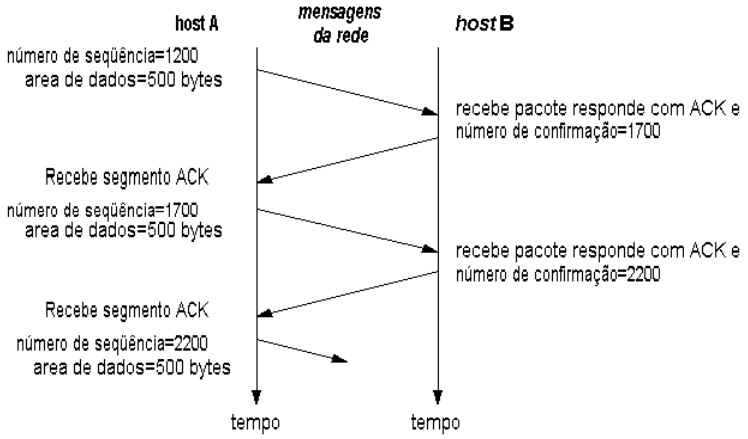

FIGURA 3.18 - Troca de datagramas entre *host* A e *host* B

A Figura 3.18 representa o envio de dados do *host* A para o *host* B, em datagramas com área de dados de 500 bytes, a partir de um número de seqüência (1200) previamente estipulado no momento do estabelecimento da conexão.

3.4.4 Retransmissão e Duplicação de Datagramas

Como visto na seção anterior, o TCP utiliza um mecanismo de confirmação de recebimento de datagramas, a fim de o receptor notificar o emissor de que a comunicação se deu com sucesso.

A Figura 3.19 representa a situação em que um datagrama é perdido na rede, no caminho até o receptor.

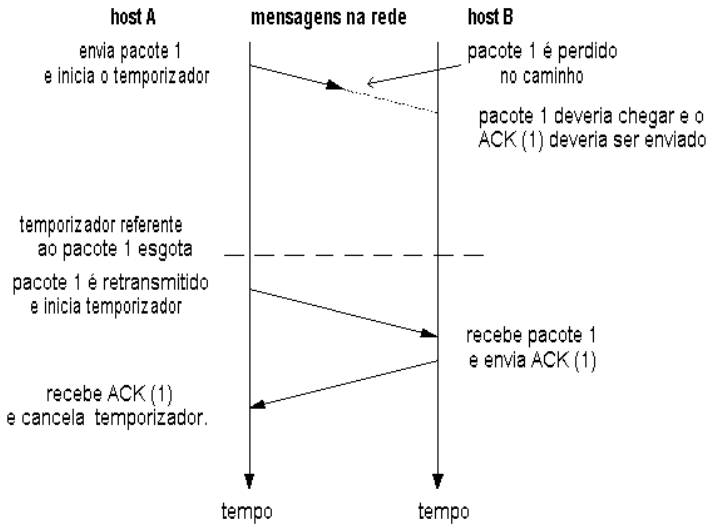

FIGURA 3.19 - Retransmissão de pacote perdido

Quando o *host* A envia o pacote, um temporizador é iniciado para aguardar a confirmação. Como o *host* B não recebe o pacote, não emite a confirmação de recebimento do mesmo. Esgotado o temporizador o *host* A, retransmite o pacote, que chega ao seu destino, é confirmado, e a comunicação procede normalmente.

Porém, é importante observar que se um pacote pode se perder no caminho do *host* A para o *host* B, poderá também ser perdido, no caminho inverso. Com isso, pode-se defrontar com a situação ilustrada na Figura 3.20.

Nesta situação, o *host* B recebeu normalmente o pacote enviado, porém a confirmação enviada para o *host* A não chegou ao destino final. O temporizador disparado pelo *host* A no momento do envio do pacote se esgota. Para ele, o pacote precisa ser retransmitido. Quando isso é feito, ocorre uma duplicação de pacotes no receptor.

Para contornar essa situação, o TCP se utiliza do número de seqüência contido no cabeçalho. Ao perceber que o pacote está duplicado, envia novamente a confirmação, mas descarta-o.

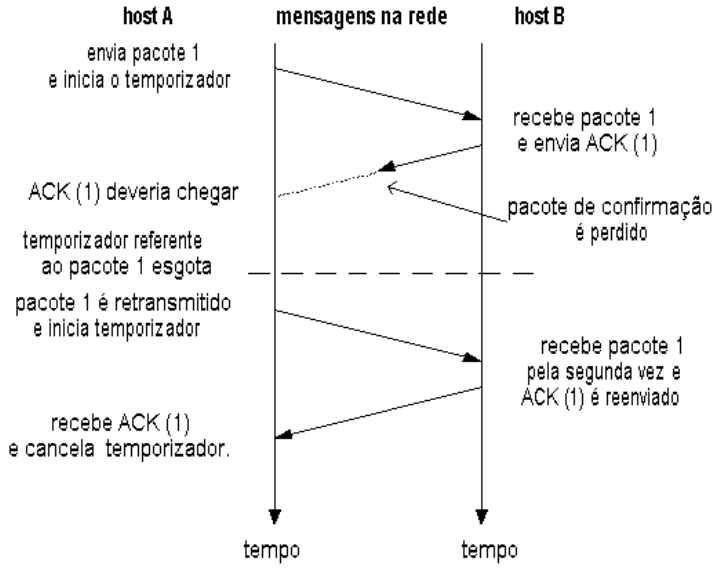

FIGURA 3.20 - Duplicação de pacote no receptor

3.4.5 Utilização de Janelas Deslizantes e Perda de Pacotes

Para melhorar o desempenho no envio dos pacotes, o protocolo TCP trabalha com o conceito de janela deslizante.. Ao invés de aguardar a confirmação de cada pacote enviado antes de enviar o seguinte, com o uso de janelas, é possível enviar vários pacotes seqüencialmente para depois aguardar a confirmação do recebimento dos mesmos. A melhora do desempenho se dá porque o tempo que seria “perdido” aguardando a confirmação é utilizado para enviar novos pacotes.

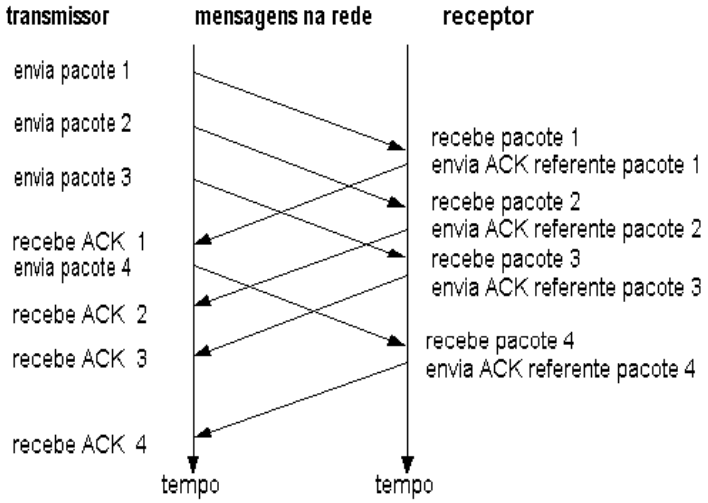

FIGURA 3.21 - O uso de janelas deslizantes

A Figura 3.21 ilustra o uso de uma janela deslizante. No exemplo utilizado, a janela deslizante tem o tamanho que equivale 3 pacotes. Observa-se que tem-se 4 pacotes a serem transmitidos, mas em nenhum momento mais de 3 pacotes estavam circulando na rede, sem confirmação.

O tamanho da janela representa o número de bytes que podem estar trafegando sem confirmação. Como o campo no cabeçalho TCP que define esse valor é formado por 16 bits, o tamanho máximo possível para a janela é de 64 KB, ($2^{16} - 1 = 65.535$ bytes). Mas normalmente esse valor não passa de 32 KB [TOR 2001].

O tamanho da janela é definido em número de octetos a serem enviados, sendo que é variável, podendo ser alterado a qualquer instante, a fim de melhorar a performance da conexão.

A Figura 3.22 representa uma janela com tamanho de 7 octetos, onde se tem três ponteiros que são utilizados para o controle dos octetos que podem ser enviados. O ponteiro (a) indica que até os octeto 1 e 2 foram enviados e confirmados. O segundo ponteiro (b), informa que os octetos 3, 4, 5 e 6 foram enviados, mas não confirmados. Entre o ponteiro (b) e o ponteiro (c) estão os octetos que ainda não foram enviados (7, 8 e 9), mas estão dentro da janela atual e podem ser enviados a qualquer momento. Os octetos à direita do ponteiro (c) estão fora da janela e não poderão ser enviados antes que a janela se mova.

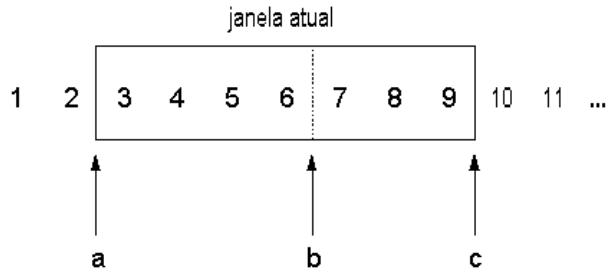

FIGURA 3.22 - Representação da janela deslizante

Deve-se observar que mesmo com o uso de janelas, o transmissor poderá ficar parado, sem enviar dados, aguardando pela confirmação de um pacote antigo. Mas mesmo sendo assim, a vantagem é grande em relação a situação inicialmente vista, onde antes de enviar cada pacote, era preciso confirmar o pacote enviado anteriormente.

Para cada pacote enviado, o transmissor dispara um temporizador para controlar o tempo de espera pela confirmação. Quando o temporizador se esgota, o pacote é reenviado.

O receptor retorna ACK para confirmar o ponto dentro do fluxo de bytes que foi recebido com sucesso. Na Figura 3.23 pode-se observar o funcionamento da janela deslizante de tamanho 6, quando um pacote intermediário é perdido. Os pacotes 1 e 2 são confirmados imediatamente. O pacote 3 é perdido. A partir deste momento, os pacotes seguintes (4, 5, 6, 7 e 8) são recebidos, mas a seqüência de bytes recebidos através deles não é confirmada, pois um conjunto de dados intermediário, referente ao pacote 3, está ausente. Isso ocorre porque a confirmação de recebimento do TCP é feita, enviando ao transmissor o número de seqüência do próximo pacote que ele espera receber.

Com a perda do pacote 3, a transmissão é interrompida, de acordo com o tamanho da janela deslizante.

Quando o temporizador do pacote 3 se esgota, então este é retransmitido. Assim que este chega ao receptor, é confirmado o recebimento da seqüência de bytes relativa até ao pacote 8. Na seqüência, os demais pacotes são enviados.

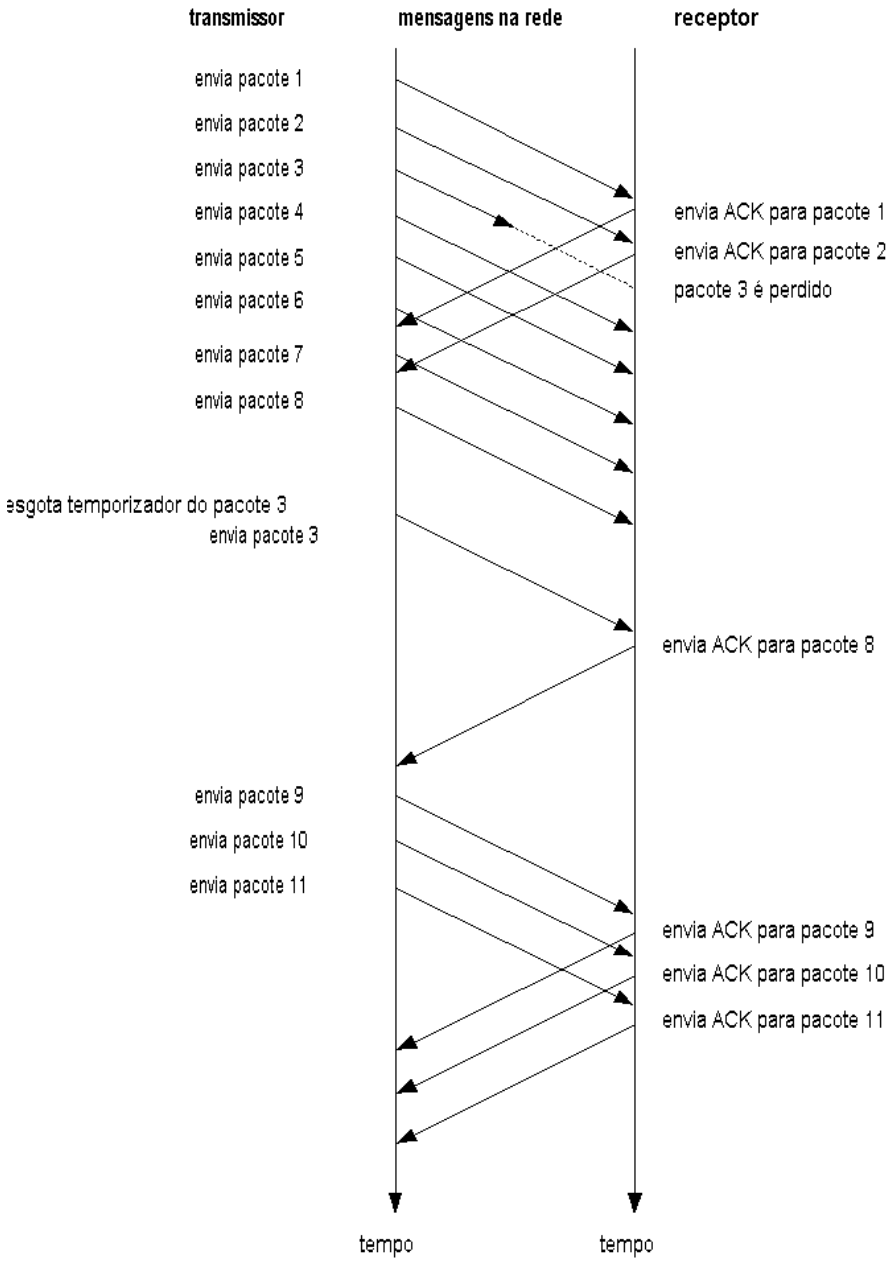

FIGURA 3.23 - Janela deslizante de tamanho 6

Na prática, quando existe a necessidade de fazer a retransmissão de um pacote, muitas vezes todos os demais da janela são retransmitidos. Isso se dá, pois o tempo de espera de confirmação de cada um deles se esgota de forma sucessiva, gerando para cada um, a necessidade de reenvio, antes da chegada da confirmação. Nesses casos tem-se uma retransmissão desnecessária.

3.4.6 Dados Urgentes

O TCP é um protocolo que faz a comunicação por meio de um fluxo, uma seqüência de bits, dividida em octetos. Porém podem ocorrer situações em que um lado quer enviar dados urgentes, de forma que estes não fiquem na fila do fluxo normal, mas que possam ser processados imediatamente pela outra entidade da conexão.

Um exemplo prático ocorre quando o usuário de um *login* remoto deseja utilizar combinação de teclas para interromper um determinado processo. Os sinais de interrupção devem enviados na forma de dados urgentes, pois precisam ser processados pelo programa da máquina remota, antes mesmo deste terminar de ler o fluxo normal de dados na fila de entrada.

Para configurar esse tipo de transmissão, o TCP faz uso do *flag* URG e do campo ponteiro urgente.

Após todos os dados urgentes consumados, o TCP informa ao programa aplicativo para que este volte à operação normal.

3.4.7 Controle de Fluxo e Congestionamento

O TCP possui também um mecanismo para o controle do fluxo de dados gerado, a fim de evitar congestionamentos. Para isso, faz uso do recurso das janelas deslizantes. As janelas utilizadas pelo TCP podem ter seu tamanho variado durante a comunicação a fim de reduzir o tráfego gerado.

Quando o *buffer* do receptor se esgota, ele solicita ao transmissor para que reduza o tamanho da janela, de forma que existam menos octetos em trânsito. Na verdade o receptor retorna ao transmissor, dentro do campo tamanho de janela, a quantidade de octetos que podem ser enviados a partir do octeto confirmado. Caso o tamanho de janela informado seja igual a zero, então, o transmissor para de enviar dados e aguarda até que o receptor lhe envie uma nova notificação informando um novo tamanho de janela, maior que zero.

3.5 UDP - *User Datagram Protocol*

Também oferece ao IP um serviço de transporte de datagramas. Quando o UDP é utilizado, ao invés do TCP, o fluxo de dados da camada de transporte continua sendo sem conexão, e ao contrário do TCP, não verifica se o pacote de dados chegou ou não ao destino. Esta é a razão que faz com que o UDP não seja utilizado para o transporte das aplicações que exijam um certo nível de confiabilidade [TAN 97] [COM 98].

A estrutura do datagrama UDP descrita a seguir, pode ser vista na Figura 3.24 [POS 80]:

- **Porta de Origem e Porta de Destino**: estes campos especificam a aplicação que originou o datagrama e a aplicação para a qual é destinada. Isto é, as portas, em cada máquina, estão associadas a uma aplicação. Analisando o conteúdo do campo porta destino, um computador, ao receber os datagramas UDP, pode saber para qual aplicação deve ser entregue o conteúdo transportado pelo mesmo.

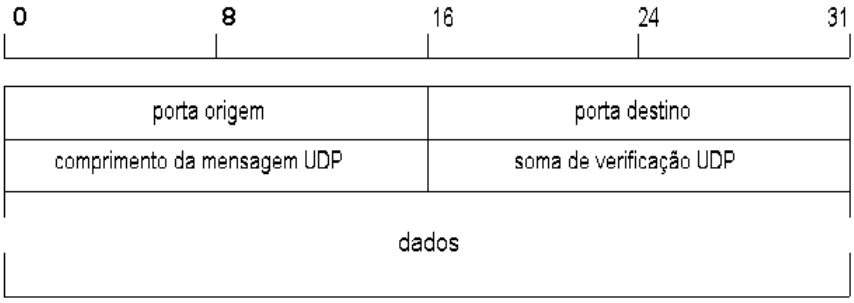

FIGURA 3.24 - Cabeçalho UDP

- **Tamanho**: especifica o tamanho do datagrama UDP, contabilizado em bytes. Esse valor inclui o cabeçalho e a área de dados. Quando não possui nenhuma informação na área de dados, este valor é igual a oito, referente ao tamanho do próprio cabeçalho.
- **Soma de Verificação**: a realização do cálculo da soma de verificação é opcional. A implementação do protocolo foi feita desta forma, a fim de evitar o *overhead* em redes que são consideradas confiáveis. Mas como esta é a única forma de se garantir que o conteúdo do datagrama está correto, ao chegar no destino, a maioria das aplicações fazem uso desta funcionalidade a fim obter maior confiabilidade. O cálculo desse valor é feito com o uso de um pseudocabeçalho, que não é transmitido, e valores presentes no cabeçalho UDP verdadeiro [POS 80]].

Como o protocolo não oferece confiabilidade, a aplicações devem se encarregar de construir mecanismos para verificar se os dados foram recebidos corretamente e também para colocar os datagramas recebidos em ordem.

Se o UDP perde no aspecto da confiabilidade, o seu lado positivo é a transmissão de dados, que é mais rápida. Isso se deve ao fato do cabeçalho ser menor e não existir a necessidade, por parte do transmissor, de aguardar a chegada da confirmação de recebimento de pacotes anteriormente enviados, para que sejam enviados novos pacotes.

Como o uso desse protocolo acarreta em responsabilidades maiores para os níveis superiores, ele é utilizado em aplicações simples, onde a taxa de envio de dados é pequena, ou a perda de pacotes não seja um problema grave. Dentre os usos mais comuns do UDP, destaca-se o SNMP (*Simple Network Monitoring Protocol*) e o DNS (*Domain Name System*).

3.6 Portas de Serviço

Serviços de rede são programas que estão sendo executados num computador, mas que podem ser acessados por outros computadores da rede. Porta de serviço são valores numéricos que variam de 0 até 65535, utilizados para identificar cada um dos diferentes serviços de rede. A porta é utilizada também para identificar os pontos finais de uma conexão feita sobre o IP.

Universalmente foi convencionado que as portas mais baixas, de 1 até 1023, também conhecidas como portas privilegiadas, ficam associadas a serviços que estão em

servidores, também conhecidos como *daemons*. Cada *daemon* fica ouvindo a chegada de pacotes da rede, esperando um que seja destinado a sua porta.

A Tabela 3.3 lista alguns dos principais serviços e as respectivas portas utilizadas. Juntamente com o número da porta, está descrito o protocolo (TCP ou UDP) associado [ZIE 2000]:

TABELA 3.3 - Portas e serviços

Serviço	Número da Porta	Protocolo
FTP	21	TCP
TELNET	23	TCP
SMTP	25	TCP
WHOIS	43	TCP
DNS	53	UDP
TFTP	69	UDP
FINGER	79	TCP
HTTP	80	TCP
POP-3	110	TCP
AUTH	113	TCP
NNTP	119	TCP
NTP	123	UDP
HTTPS	443	TCP

Já as portas de numeração maior, as não privilegiadas que variam de 1024 até 65535, são utilizadas pelo cliente da conexão.

É importante salientar que esta convenção criada para a numeração das portas não oferece nenhuma garantia. Isto significa que não se pode confiar cegamente que atrás de determinada porta será encontrado o serviço respectivo, conforme a literatura prega. Pode ocorrer de se ter um serviço X rodando atrás de uma porta que convencionalmente deveria ser de um serviço Y, pois isso pode ser configurado no servidor.

Ao tentar conectar-se com um determinado serviço, o cliente indica a porta de respectiva. Se o serviço não estiver disponível, nada acontecerá. A solicitação chega até a máquina destino, mas nenhum programa estará recebendo os pacotes na porta específica. Caso exista um programa aguardando dados na porta referenciada, porém não seja o programa que o cliente espera encontrar, então ocorrerá um erro na aplicação. Se tudo estiver correto, a comunicação se dará conforme a especificação do protocolo em uso.

3.7 Cuidados com o uso do TCP/IP

O TCP/IP e a Internet estão sempre tão associados que muitas vezes se confundem e são tomados como um único. Na verdade, a Internet surgiu e se desenvolveu baseada nos protocolos da família TCP/IP.

O projeto inicial tinha como objetivo fornecer conectividade entre os computadores para uma comunidade restrita de usuários, que confiavam mutuamente entre si. Posteriormente, quando a Internet saiu desse meio restrito, algumas problemas

foram detectados. Problemas esses relacionados ao principal protocolo utilizado, o IP (protocolo internet) [TAN 97].

As deficiências do IP estão associadas ao fato de que este não foi projetado originalmente para oferecer segurança. A preocupação maior era contornar falhas de *hardware*, mas dentro de um ambiente confiável. Qualquer nível maior de segurança ficou sob responsabilidade dos protocolos de nível superior.

Segundo [GON 2000], segurança não é a característica principal do TCP/IP. Por isso, tem-se procurado desenvolver dispositivos a fim de adicionar segurança às redes, principalmente quando esta estiver conectada com a Internet.

Um dos primeiros e mais simples recursos utilizados foi a implantação de listas de acesso nos roteadores que dividem as redes [HEL 99]. Mais tarde outros mecanismos foram implementados a fim de proporcionar um nível melhor de segurança.

Segundo [WEB 2000], os mecanismos mais empregados para implementar a segurança são:

- Métodos criptográficos.
- Mecanismos de autenticação;
- *Wrappers;*
- *Firewalls;*
- Ferramentas de detecção de falhas;

3.7.1 Métodos Criptográficos

Para proteger as informações que trafegam pela rede, para que não sejam lidas por intrusos, são aconselhados métodos de criptografia. O objetivo é simples, embaralhar os dados para que eles somente possam ser entendidos pelos verdadeiros destinatários.

3.7.2 Mecanismos de Autenticação

Busca métodos aperfeiçoados para a realização da autenticação, que normalmente é feita com base em senhas, com o método: “algo que o usuário saiba”. Outras soluções, porém de custo mais elevado, oferecem maior segurança ao sistema. São mecanismos que implementam senhas para utilização única, isto é, mesmo que esta seja capturada por algum intruso, ela não terá mais validade no momento em que ele for tentar fazer uso dela. Pode-se obter um detalhamento maior sobre o assunto em [HAR 96].

O método que faz uso de “algo que o usuário possua”, que envolve a autenticação biométrica (leitura de digitais, retina, traços da face,...) também se enquadram neste grupo, porém ainda não estão tão difundidos.

3.7.3 Wrappers

São programas que encapsulam serviços da rede. Funcionam como um desvio. Quando um serviço é solicitado, ao invés de ser ativado o serviço original, o wrapper é ativado. Este pode efetuar uma série de operações e depois sim, passar o controle ao serviço original. Com isso, pode-se implementar características especiais desejadas, tais como: realização de um registro de ocorrências (log) paralelo e mais detalhado.

3.7.4 Firewalls

"*Firewall* é um componente ou um conjunto de componentes que restringem o acesso entre a rede protegida e a Internet ou outra rede" [CHA 95]. A idéia básica do *firewall* é proteger uma rede específica (privada) de acessos considerados maliciosos, originados em outras redes. Para o ter sucesso, deve ser implementado num ponto estratégico da rede, que seja um caminho único de comunicação da rede que se quer proteger, com o mundo externo.

Aproveitando as características de restrição que possui, além de controlar o acesso para a rede "protegida", o mecanismo de *firewall* pode ser utilizado para restringir o acesso dos usuários da rede interna, para outras redes.

Em função de suas características de funcionamento, um *firewall* nada poderá fazer para proteger a rede de ataques originados internamente, uma vez que só tem controle sobre o tráfego que passa por ele.

A função mais conhecida de um *firewall* é a filtragem de pacotes. Na filtragem ocorre uma seleção dos pacotes que podem seguir seu fluxo normal. As informações contidas no cabeçalho dos datagramas, como endereço e porta origem e destino, são analisadas e comparadas com as restrições impostas ao tráfego em ambos os sentidos. Caso tal tráfego seja permitido, então o pacote é roteado para a outra rede e segue normalmente ao destino, caso contrário é barrado.

Segundo [HEL 99], um *firewall*, além de executar listas de acesso, permite diversas outras funções relacionadas à segurança, tais como:

- inspeção de informações de estado;
- serviços de *proxy*;
- criptografia;
- autenticação;
- geração de alertas.

a) Inspeção de informações de estado

Em um fluxo de datagramas que possuem o mesmo endereço de origem e destino, podemos ter uma tentativa de *logon* repetida. Isso pode indicar uma tentativa de ataque. A inspeção de informações de estado, é na verdade, uma verificação feita no conteúdo dos pacotes, e, quando uma situação suspeita for detectada, dependendo da configuração do *firewall*, os pacotes seguintes poderão ser barrados.

b) Serviços de *proxy*

O *proxy* é um intermediário entre o solicitante de um serviço e o serviço desejado. Os pacotes não trafegam livre e diretamente através do *firewall*; eles são inspecionados e depois sim, repassados ao destino.

No caso de um *proxy* FTP, por exemplo, permitirá que seja feito um controle sobre a utilização do mesmo. Um bom exemplo é o bloqueio do uso do comando MGET. Isso poderia ser feito para evitar que ao tentar fazer um *download* de vários arquivos com este comando a comunicação com a internet fique saturada por um longo período.

c) Criptografia

Como um recurso mais recente, o firewall tem recebido a função de criptografia seletiva. Isso permite que dados que tenham destinos pré-selecionados sejam criptografados, antes de serem enviados para a internet, enquanto outros permaneçam inalterados. Com este recurso pode-se criar um a espécie de um túnel lógico através da estrutura física da Internet, para conectar computadores distantes geograficamente. É o que se pode chamar de rede privada virtual (VPN) [HEL 99].

d) Autenticação

A autenticação possibilita que o *firewall* possa definir se um acesso é ou não permitido, baseado também no usuário da conexão e não somente nas informações contidas no cabeçalho, como informações de endereço e porta origem e destino. Pode permitir acesso a certos computadores e serviços para usuários privilegiados, enquanto que para outros será negado [GON 2000].

A autenticação pode ser feita com a utilização de *tokens* criptografados, mecanismo de *one-time password*, mas o que mais é utilizado ainda são as velhas *password* de texto simples, que são as menos seguras.

e) Geração de alertas

Geração de alertas, como o próprio título informa, são utilizadas para avisar alguém, normalmente o administrador do sistema, sobre alguma ocorrência. Por exemplo, cada tentativa de *login* fracassada pode gerar um aviso, em função de ser uma suspeita de invasão.

3.7.5 Detecção de Falhas

As máquinas que estão conectadas na rede, principalmente os servidores, apresentam mecanismos de registro de atividades. São os chamados *logs*, que registram o histórico de ocorrências sob determinado aspecto do comportamento da máquina. Com a análise destes *logs*, é possível se identificar possíveis falhas no comportamento de usuários ou ameaças à segurança da rede.

Por si só, este recurso é limitado e exige um trabalho cansativo por parte dos administradores. Na busca de melhores condições de detecção foram desenvolvidos grupos de ferramentas, dentre os quais destacam-se:

a) **Ferramentas de análise**: verificam as configurações das máquinas conectadas na rede, em busca de falhas na configuração;
b) **Verificadores de integridade**: são utilizados para controlar alterações dos arquivos armazenados em um computador;
c) **Verificadores de senhas**: tem a função de verificar e controlar as senhas utilizadas pelos usuários, não permitindo que estes façam uso de senhas fáceis ou óbvias, que possam ser facilmente descobertas.
d) **Analisadores de *log***: são ferramentas criadas com o intuito de auxiliar os administradores na tarefa de estudar os registros de ocorrência de um computador,
e) **Sistemas de detecção de intrusão**: visam identificar uma invasão durante a tentativa, antes que ela se consuma de fato. O objetivo é gerar uma reação contra a ação do atacante, no menor tempo possível. Para fazer a identificação da tentativa de intrusão, dentre outros métodos, destaca-se o

monitoramento do tráfego de rede. Através da análise do trafego de pacotes, são identificadas ações suspeitas.

4 Monitoramento da Rede Ethernet

Vistas as características do TCP/IP e os principais mecanismos empregados a fim de implementar segurança nesse ambiente, observa-se que em alguns destes, o monitoramento da rede tem grande importância.

Como foi estudada, a utilização de *firewalls* para proteger as redes privadas quando conectadas à Internet é indispensável. Este mecanismo permite controlar quais serviços remotos os usuários locais podem acessar, bem como, que tipo de acesso os usuários externos podem fazer dentro da rede privada. Pode ainda, ser aplicado para controlar acesso entre duas redes privadas, ou em relação a uma subrede [TAN 97] [GON 2000].

O que se espera de um mecanismo desse tipo, é que além de cumprir a sua tarefa de negar ou permitir o tráfego entre as duas partes, ele possa gerar informações sobre a situação do ambiente em um dado momento. Na verdade, essa informação não é obtida de forma direta. Arquivos de *log* são gerados, montando um histórico das ocorrências. Alguns *logs* relatam os casos em que certas tentativas de acesso foram negadas. Já outros, as trocas de informações realizadas [BRE 99].

Mesmo com todos estes dados em mãos, o caminho para se chegar a informação final desejada pode ser longo e árduo. Para que algo possa ser concluído é preciso que se analise, compare e que se volte a analisar diferentes arquivos de *log* gerados pelas diversas partes do mecanismo de *firewall*. Não bastando somente a dificuldade com a quantidade de arquivos, tem-se ainda problemas com a extensão que estes "históricos" podem ter. Em ambientes que não podem ser chamados de pequenos, o tamanho destes arquivos pode ser desagradavelmente grande.

A tarefa acima descrita pode ser classificada como um trabalho complexo e braçal. Em virtude disso, é difícil chegar a uma conclusão rapidamente, e os resultados atingidos poderão não ter a precisão esperada. Em ocasiões em que é necessário se obter informações instantâneas sobre determinada situação da rede, deve-se então partir para a utilização de outras formas de verificação.

Um monitor de transações dos serviços internet se torna uma ferramenta interessante para esta finalidade. Este, observando o tráfego que passa por um determinado segmento, poderá rapidamente concluir, ou oferecer informações que facilitem uma tomada de decisão por parte do administrador da rede.

Também quando se fala em sistemas de detecção de intrusão, pode-se ver a importância de um monitor desse tipo.

A seguir estão descritas algumas situações em que um monitor pode se apresentar como uma ferramenta de grande utilidade, dentro de cada contexto:

4.1 Para verificar a eficiência do *firewall*

Quando um *firewall* é configurado, não é possível se ter garantias de que ele está funcionando de acordo com o esperado e planejado [SIY 95] [GAR 96].

Quando se projeta um mecanismo de *firewall*, faz-se inicialmente um planejamento teórico do que pode e do que não pode trafegar na rede, ou entre redes. Com base neste planejamento é feita a implementação propriamente dita. São configurados os equipamentos que irão realizar o trabalho do *firewall*. Porém, problemas diversos podem gerar distorções nos resultados esperados. As causas podem ser erros na configuração de software ou até mesmo, no projeto de *hardware*.

Um monitor de transações dos serviços internet pode ser utilizado como ferramenta de aferição. Isto é, possibilita que seja comparado o que está realmente trafegando, com o que foi permitido no planejamento teórico previamente realizado. Ele poderá servir de apoio para verificação do cumprimento das políticas de segurança definidas para a rede.

4.2 Para orientar a reconfiguração automática de um *firewall*

A idéia é semelhante à que foi descrita no primeiro item, diferenciando-se, no entanto, na profundidade do projeto. Pode-se, partindo-se da utilização de uma ferramenta de monitoramento, projetar um sistema de configuração dinâmica de *firewalls*. Tal sistema poderia ser baseado em prioridades na utilização da rede, por exemplo. As prioridades poderiam ser definidas em função do tipo de serviço, ou das entidades origem ou destino da conexão. Isto é, o monitor faz o controle da utilização da rede de modo a obter informações sobre os tipos de serviços utilizados e por quais entidades.

Neste exemplo simples, políticas podem ser definidas para que quando o tráfego de rede esteja acima de um valor "X" predefinido, alguns serviços considerados supérfluos sejam bloqueados. E no processo inverso, quando a taxa de utilização for reduzida, voltassem a serem liberados tais serviços.

Nesse caso, um monitor de transações dos serviços internet teria a função de gerar uma base de informações sobre o tráfego, que seriam utilizadas para disparar alarmes responsáveis por alterações no sistema de segurança.

Esta idéia pode, em primeiro momento, parecer estranha pois o monitoramento do tráfego da rede seria possível de ser feito, por meio da coleta de informações nos equipamentos de rede. Alguns equipamentos para rede Ethernet, como *hub* repetidores e *hubs* de comutação (conhecidos simplesmente como *hub* e *switchs* respectivamente), podem armazenar informações sobre o tráfego que passa por eles. São os equipamentos providos de recursos de gerenciamento.

Porém, é importante salientar ainda que, nem todos os equipamentos de rede apresentam as características de gerenciamento. Principalmente por questões de custo, muitas vezes são utilizados *hubs* não gerenciáveis [TOR 2001] [TAN97].

Mesmo os *hubs* e *switchs* que apresentam recursos de gerenciamento, atuam somente sobre as camadas mais baixas do TCP/IP. Através desse tipo de monitoramento é possível, por exemplo, saber como está a taxa de utilização da rede, ou ainda, quais as portas que possuem altas taxas de tráfego de entrada ou saída. Mas um controle sobre endereços IPs ou sobre serviços utilizados não é possível. Para obter tal controle deve-se utilizar recursos externos ao equipamento.

Uma ferramenta que faz monitoramento das transações internet na rede pode analisar dados dos níveis mais altos da topologia, permitindo contabilizar a utilização da rede, em função de uma aplicação ou de um *host* específico.

Saindo do contexto de *firewalls* e passando para outro mecanismo empregado para proporcionar maior segurança à rede, os detectores de falhas, também pode-se observar aplicações para um monitor de tráfego de rede.

4.3 Monitor para Sistema de Detecção de Intrusão

Conforme visto na seção 3.5, Sistemas de Detecção de Intrusão (SDI) podem fazer uso do monitoramento de tráfego da rede para identificar ações consideradas

suspeitas. Segundo [NAK 2002], os SDI podem ser classificados em 3 grandes grupos, segundo a forma como fazem o monitoramento:

a) SDI baseado no monitoramento do *host*;
b) SDI baseado no monitoramento da rede;
c) SDI híbrido.

O primeiro, baseado no monitoramento do *host*, age com base nas informações de arquivos de *log* ou de agentes de auditoria; entre outras atividades, monitora o acesso e alterações em arquivos do sistema, modificações nos privilégios de usuários, monitora também processos e programas que estão sendo executados, faz checagem de integridade dos arquivos, etc.

O segundo monitora o tráfego do segmento da rede, onde a detecção é realizada por meio da captura e análise dos pacotes, que são comparados com padrões e assinaturas conhecidos.

O SDI híbrido, como a própria denominação sugere, é uma mescla dos outros dois, vistos anteriormente. Aproveita os pontos positivos de cada um, para formar um SDI mais completo e eficiente.

De qualquer forma, tanto um SDI puramente baseado em rede como um SDI híbrido, utilizam sensores ou agentes no segmento de rede que será monitorado. O sensor é o componente responsável pela captura de pacotes do segmento de rede, e também faz a formatação de alguns dados de acordo com o projeto do mecanismo de detecção.

Este estudo não tem o objetivo de construir um monitor completo voltado para este tipo de aplicação, cobrindo todas as necessidades do mesmo. Porém, ao final deste estudo, a técnica de captura e manipulação de pacotes que trafegam na rede estará dominada, servindo de base de partida para futuros trabalhos.

4.4 Monitor para auxiliar na tomada de decisões

Um monitor que apresente as conexões TCP/IP ativas na rede e os serviços utilizados, é útil como uma ferramenta de apoio na tomada de decisões por parte do administrador da rede.

Em determinadas situações, o administrador tem a necessidade de fazer verificações sobre o comportamento do tráfego, seja em função de um protocolo específico, ou então de *host* em particular.

Um monitor proporciona maior flexibilidade e instantaneidade na busca de informação. A flexibilidade é obtida pelo fato do monitor permitir que a pesquisa, ou a busca de informações seja adaptada, conforme a necessidade. Isto é, por meio de filtros, ele pode fornecer uma focalização em aspectos específicos que o administrador da rede esteja querendo observar. A instantaneidade provém da característica de exibir o que está acontecendo no momento, diferentemente de *logs*, que armazenam um histórico de ocorrências para posterior verificação.

A utilização do monitor, com a finalidade de tomada de decisão, pode ocorrer, por exemplo, quando se inclui um novo roteador na rede. Com o monitor, pode-se verificar a ocorrência de mensagens ICMP de redirecionamento. O comportamento da rede poderá indicar se a situação está normalizada ou se algo precisa ser mudado.

4.5 Formas de monitorar a rede

Para realizar a captura de pacotes necessários para a realização do monitoramento da rede Ethernet, tem-se duas alternativas, que diferenciam-se na sua forma básica de ação.

Na primeira opção, aqui denominada de monitoramento de pacotes por passagem, o programa responsável por coletar os dados da rede deve estar colocado em uma posição estratégica dentro da estrutura física da mesma. Deve estar localizado num ponto que seja caminho único para os dados da rede, isto é, todos os dados que saem ou entram da rede, devem passar por ele..

Outra alternativa, é a captura de pacotes que trafegam no meio físico, fazendo-se uso de uma placa de rede trabalhando em modo promíscuo.

A seguir serão detalhadas ambas opções vistas acima.

4.5.1 Monitoramento de pacotes por passagem

Nesta opção, deve existir uma única saída para o fluxo de dados, conforme ilustrado na Figura 4.1. A estrutura física é montada de modo que todo o tráfego da rede tenha uma saída única e obrigatória para os pacotes. Nesse ponto da rede, é feita a análise dos dados que trafegam e com isso pode-se obter a "imagem" das comunicações que estão se realizando com a rede.

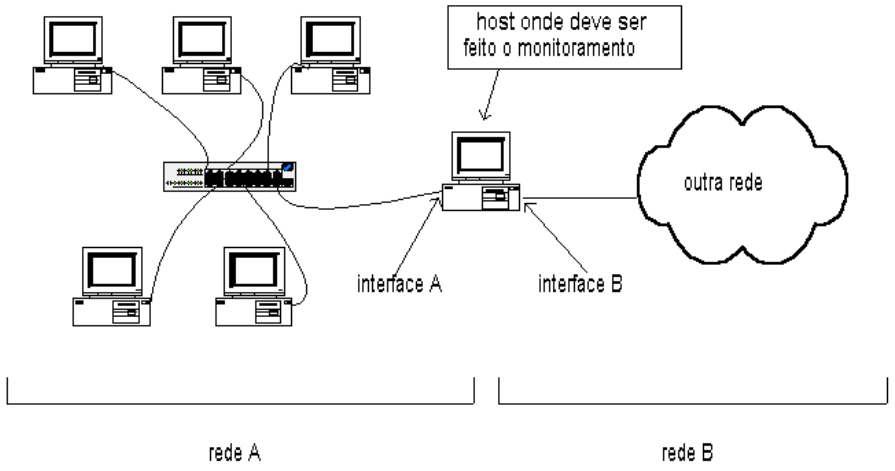

FIGURA 4.1 - Monitoramento de pacotes por passagem

Na verdade, este é um mecanismo semelhante aos filtros de pacotes. Exige que o *host* no qual o software de monitoramento será hospedado, tenha duas interfaces de rede. Uma para a rede a ser analisada, aqui chamada de rede A (interface A) e outra para a rede externa, que pode ser a Internet, denominada rede B (interface B).

O princípio do funcionamento deste mecanismo diz que ao serem direcionados de uma rede para outra (de uma porta para outra), os pacotes são analisados pelo software de monitoramento. Com isso, todo o tráfego entre as redes é registrado.

O monitor não tem nenhuma função de censurar tráfego, mas somente “olhar” os dados que estão passando, enquanto estes trafegam livremente. Pode ser implementado por meio de alterações feitas sobre um sistema de filtro de pacotes tradicional, de modo a coletar e exibir as informações coletadas, oferecendo maior interatividade com o seu usuário.

A grande vantagem deste método é que todos os pacotes que fizerem a passagem de uma rede para outra, serão registrados, mesmo em situações de grande tráfego.

Porém, analisando o funcionamento deste mecanismo, pode-se detectar algumas desvantagens:

- São monitorados somente os quadros que saem ou chegam na rede controlada, no caso a rede A. Isto é, quadros que passam pelo computador que está realizando o monitoramento. Desta forma, não é garantida a visualização da comunicação que pode estar ocorrendo internamente, entre as máquinas em cada uma das redes A e B;
- Esse método também apresenta uma certa complexidade para implementação. Para se poder monitorar uma rede, é exigido que se tenha um ponto obrigatório de passagem dos dados.

  Para se poder monitorar uma rede simples, onde se tem clientes e servidores em um mesmo segmento, dentro de um mesmo domínio de colisão, é preciso promover alterações na estrutura física da mesma.
- O mesmo ocorre quando existe um ambiente que possui dois segmentos de rede. É preciso alterar a estrutura, implantando um computador que servirá de passagem única entre os dois segmentos.
- A exigência da implementação do ponto de passagem dos pacotes não é boa, pois retira do mecanismo a sua transparência em relação ao usuário. Não se trata de um mecanismo que pode ser ativado ou desativado a qualquer momento, sem interferir no funcionamento da rede monitorada. Estas mudanças interferem a ponto de exigir reconfiguração do *software* de rede dos *hosts* a ela conectados.
- Com a exigência da criação do ponto obrigatório de passagem dos pacotes, é criado mais um salto no caminho até o destinatário. Com isso, tem-se um acréscimo no tempo de viagem dos pacotes até seu destino. Sob o ponto de vista do congestionamento, tem-se a possibilidade de ocorrência de um gargalo na rede. Quanto ao aspecto da segurança, está-se incluindo mais um ponto de falha.

4.5.2 Monitoramento em Modo Promíscuo

Como estudado anteriormente, dentro dos domínios de colisão das redes Ethernet, os quadros chegam a todas as interfaces conectadas ao meio de transmissão, independente de quem seja o destinatário.

Uma interface de rede operando em modo normal lê algumas informações de cada quadro que trafega na rede, porém não o lê por completo. A interface analisa até o campo de endereço ethernet destino, se este não corresponder ao próprio endereço ethernet da interface ou a um dos endereços *multicast* ou *broadcast* que a interface está programada para receber, o restante do quadro será ignorado. Caso corresponda, o quadro é processado por completo, sendo repassado para os protocolos das camadas superiores [SPU 2000].

Portanto, se a interface for configurada para aceitar todos os quadros que passam pelo meio de transmissão ao qual está conectada, pode-se fazer o monitoramento do que está acontecendo com a rede.

Para isso, utiliza-se um computador com uma interface de rede operando no modo promíscuo. Com este recurso, todos os quadros que trafegam no meio físico são repassados para o aplicativo que está sendo executado no computador, de forma diferenciada do funcionamento normal da interface, onde cada computador somente processa os dados a ele destinados.

O software que coloca a interface em modo promíscuo e processa os quadros que passam pela rede, é conhecido como *sniffer*.

Comparado com o método do monitoramento de pacotes por passagem, pode-se citar vantagens do modo promíscuo:

- Através deste método, é possível analisar todas as comunicações que estão sendo realizadas entre os computadores pertencentes ao domínio de colisão. Seja entre si, ou para com computadores pertencentes a outros domínios de colisão, ou ainda para com computadores pertencente a outras redes, conforme Figura 4.2.

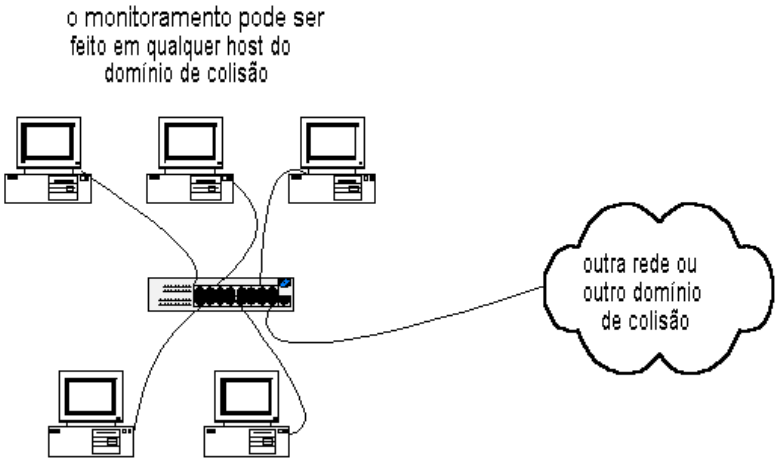

FIGURA 4.2 - Monitoramento em modo promíscuo

- Este método é transparente aos usuários e aplicações da rede. O software utilizado para monitorar não tem outra função na rede, senão simplesmente a de coletar informações sobre o tráfego, de forma passiva. Não faz parte da estrutura mínima necessária para o funcionamento da rede. Isto é, pode ser ativado e desativado a qualquer instante, sem implicar em qualquer mudança no comportamento normal da mesma.
- Em função da característica acima, o ponto de monitoramento dentro da rede pode ser alterado de posição. É possível ainda, se fazer o monitoramento em múltiplos pontos na rede, de forma transparente.
- Como nessa estrutura não existe a necessidade de um ponto concentrador do tráfego, o risco de formação de gargalo para a rede no computador que está executando o processamento dos pacotes monitorados, não existe.

Se o monitoramento utilizando a interface em modo promíscuo apresenta esta série de vantagens, por outro lado, existe o risco de que pacotes passem despercebidos pelo programa de monitoramento. Isto é, em situações de grande tráfego, a interface poderá não ser capaz de processar todo o fluxo de pacotes que passa pelo meio de transmissão. Essa situação se torna mais crítica quando for freqüente ou prolongada.

4.6 Softwares para o monitoramento

Como visto na seção anterior, fazer a captura de pacotes por meio de uma interface de rede que opera em modo promíscuo, é a forma mais transparente ao usuário de monitoramento de uma rede Ethernet.

Para realizar esse tipo de monitoramento, existem no mercado inúmeras ferramentas. Tais softwares, que são genericamente chamados de *sniffers*, ou programas *snif*.

Porém "*sniffer*" é uma marca registrada pela Network Associates que referencia um produto chamado "Sniffer® Network Analyzer" [GRA 2000].

Neste estudo, tais ferramentas serão denominadas de programas de captura de pacotes.

Programas de captura de pacotes podem ser utilizados para duas grandes finalidades distintas:

- Capturar, de forma ilícita, informações que trafegam na rede;
- Proporcionar suporte para redes, como uma ferramenta de apoio aos administradores.

Indivíduos de caráter duvidoso fazem o primeiro uso citado, para capturar identificadores de usuários, senhas, números de cartões de crédito, ou simplesmente para bisbilhotar as comunicações feitas através da rede.

Mas o propósito deste estudo está focalizado no segundo tipo de atividade citado. O desejo existente é de se contribuir com a comunidade a fim de se obter ferramentas que facilitem as tarefas de administração das redes de computadores.

O comentário anterior é importante, pois algumas das ferramentas que serão citadas a seguir, apresentam características que facilitam o seu uso para atividade ilícitas. Mesmo que estas opções sejam citadas, não está sendo feito aqui, um incentivo para tal uso. Mas sim, tal fato poderá ser encarado como um alerta para quem está preocupado em preservar as informações.

A captura de pacotes na rede é tipicamente utilizada, com boas intenções, para:

- Verificação de problemas de comunicação;
- Análise de performance da rede;
- Detecção de intrusão;
- Gerar um registro (*log*) paralelo do tráfego da rede de forma não prevista pelo atacante;
- Como uma ferramenta de ensino e aprendizagem de redes de computadores.

Segundo [GRA 2000], os principais componentes de um programa de captura de pacotes são:

a) *Hardware*:

> Refere-se aos adaptadores de rede que utilizados. A maioria dos produtos utilizam um *hardware* padrão, que permite a simples captura de dados. Entretanto, para que seja possível se fazer

verificações mais complexas, como: análise de erros de CRC, problemas de voltagem, problemas de cabeamento, erros de negociação, etc, é preciso utilizar adaptadores especiais.

b) *Driver* de Captura:
   Este componente, que é de grande importância, é o responsável por apanhar os dados que chegam no adaptador e colocá-los no buffer específico. Antes, porém, é interessante que faça uma seleção (filtragem) dos pacotes capturados.
c) *Buffer*:
   Depois de capturados, os pacotes vão para o buffer. Aqui existem dois modos para o tratamento dos dados:
   - pacotes são armazenados no buffer enquanto ele não estiver cheio, caso contrário, são descartados;
   - pacotes novos sobrepõem os pacotes mais antigos no buffer, quando este estiver no seu limite.

   O segundo método é mais indicado para situações em que se está mantendo um registro do tráfego em disco. Neste caso, ocorrerá uma rotação, de forma a manter o arquivo de saída dentro do seu limite de crescimento.
d) Mecanismo para análise em tempo real:
   Diz respeito a análise dos pacotes enquanto estes estão trafegando na rede. Permite fazer o processamento imediato dos dados capturados. As aplicações mais indicadas são para detecção de problemas ou falhas na rede.
e) Mecanismo para decodificação:
   Responsável por identificar os pacotes capturados, quanto ao protocolo específico ao qual foi construído. Permite que sejam acessadas informações que estão armazenadas em cada campo do protocolo. Através desta análise, é possível se entender a função do pacote dentro do contexto da comunicação sendo executada.
f) Injeção de pacotes:
   Mesmo que fuja um pouco do conceito inicial, que é de capturar pacotes, alguns produtos apresentam características específicas que permitem criar pacotes manualmente ou automaticamente, e enviá-los pela rede.

Existem programas para captura de pacotes construídos para trabalhar sobre os diversos sistemas operacionais. Aqui, serão destacadas ferramentas construídas para a plataforma linux. Como a quantidade oferecida é grande, neste estudo foram selecionadas algumas, cuja seleção se deve aos seguintes critérios, citados em ordem decrescente de prioridade:

a) Deve ser executada sobre linux e ser um *software* livre.
b) Deve ser referenciada em bibliografias da área [HAT 2002] ou sites relacionados à captura de pacotes e segurança. Os sites consultados foram:
   - www.packetstormsecurity.com
   - www.securityfocus.com;
c) A ferramenta deve possuir uma página na internet, com informações sobre o desenvolvimento e o uso da mesma;

d) Preferencialmente, o grupo de desenvolvimento da ferramenta deve estar ativo, disponibilizando atualizações recentes;

Dentro destes critérios, foram selecionadas algumas ferramentas, com características variadas, a fim de se levantar as opções existentes.

4.6.1 darkstat

Esta é uma ferramenta simples, que gera saída para a tela texto e para a tela gráfica [DAR 2001].

Na opção texto, mediante o uso da opção *verbose*, permite a visualização de algumas informações dos pacotes que estão trafegando pela rede.

A função mais interessante, porém, se dá mediante a disponibilização das informações estatísticas coletadas, através de um servidor *web* oferecido. Fazendo uso de um navegador *web*, é possível se visualizar uma série de gráficos que dizem respeito ao tráfego dos dados, classificados por *host* (permitindo identificar os maiores geradores de tráfego), por portas de serviço, ou por protocolos (da família TCP/IP).

Esta ferramenta utiliza libpcap para captura de pacotes, porém não faz uso de suas facilidades para a filtragem de pacotes.

4.6.2 ntop

É semelhante ao darkstat, porém mais completo. Apresenta uma gama maior e mais completa de gráficos estatísticos sobre o tráfego TCP/IP na rede. De forma igual também, carrega um servidor *web* próprio, a fim de disponibilizar acesso remoto às estatísticas coletadas [NTO 2001].

Apresenta opções para processar pacotes obtidos a partir de um arquivo (*dump*) gerado com tcpdump. Também permite que seja gerado um arquivo com o mesmo formato do que é gerado pelo tcpdump. Baseado na libpcap, ntop também aceita expressões de filtragem padrão desta biblioteca.

4.6.3 tcpdump

O tcpdump é muito próximo à libpcap. Diz-se isso, pois foi concebido pela mesma equipe que construiu a biblioteca. É uma ferramenta que permite visualizar o fluxo de tráfego na rede (dump). Pode ser utilizado para visualizar as informações dos cabeçalhos dos pacotes que passam pela interface de rede, e que correspondem à expressão de filtragem previamente definida [TCP 2001]. Os recursos de filtragem oferecidos são avançados, permitindo montagem de expressões sobre bits dos cabeçalhos.

Apesar de parecer uma ferramenta simples, sem interface avançada, o tcpdump serve como base para vários sistemas de detecção de intrusão, coletando informações que servirão posteriormente, como entrada para tais ferramentas [HAT 2002]. Assim como pode gerar tais arquivos, tem capacidade de receber os mesmos como entrada também.

4.6.4 snort

O snort vai além de um programa para capturar e registrar pacotes [SNO 2001] [CAR 2001]. É uma ferramenta que realiza análise de protocolo, fazendo pesquisas e comparações a fim de detectar ataques ou tentativas de ataques na rede. Para esta tarefa

de detecção, possui uma linguagem que permite a criação de regras que formam o mecanismo de detecção.

Esta ferramenta possui também a opção *verbose*, que é utilizada para escrever na console, informações dos pacotes IP que estão trafegando na rede. As informações exibidas, dizem respeito aos cabeçalhos IP, TCP, UDP, ICMP ou ARP.

Fazendo uso da biblioteca libpcap para a captura de pacotes, o snort incorpora as facilidades de filtragem da mesma, e pode processar informações obtidas a partir de um arquivo (*dump*) no formato tcpdump, previamente gerado.

4.6.5 sniffit

O sniffit foi construído utilizando ncurses, e é uma ferramenta que tem como principal característica o monitoramento das conexões TCP ativas. Para cada conexão ativa, são exibidas informações de endereço e porta de serviço associados, para ambas as entidades. Selecionando uma conexão, é permitido monitorar os dados que estão trafegando por ela.

Mesmo sendo baseado na libpcap, o sniffit é pobre nas possibilidades de filtragem apresentadas, possibilitando apenas filtragem por endereço e porta, origem e destino [SNI 2001].

4.6.6 ettercap

Etthercap é uma ferramenta que além de operar sobre uma rede conectada com um hub, possui a capacidade de fazer captura de pacotes em uma rede com switchs. Para tal, faz uso da técnica denominada *man-in-the-midle* [ETT 2001].

Por meio desta técnica, a máquina atacante fica no meio da comunicação entre as duas entidades vítimas. Isso se dá porque o ettercap engana as vítimas por meio de resposta ARP forjadas. Através delas, informa o seu próprio número MAC a cada uma das entidades vítimas, fazendo-se passar, para cada uma delas, pela outra extremidade da comunicação. Desta forma, as duas entidades alvo enviam seus pacotes para o atacante, que depois os repassa aos verdadeiros endereços.

Apresenta a possibilidade de visualizar o tráfego na console, similarmente ao tcpdump, mas a interface principal é feita utilizando ncurses. Com ncurses são exibidas as conexões detectadas no monitoramento de um domínio de colisão (conectada a um *hub*), identificando a situação da mesma. Isto é, informa se a conexão está ativa ou fechada. Permite visualizar o tráfego em cada conexão, bem como destruí-la. Possibilita também a montagem, de forma manual, de pacotes e o seu envio pela rede.

Ettercap é capaz também de coletar informações de usuários e senhas que estão trafegando na rede, para alguns protocolos, como TELNET, FTP, POP, RLOGIN, SSH1. Permite também gerar arquivos *dump* no formato pcap e ler arquivos previamente gerados, por ferramentas compatíveis, como tcpdump.

A ferramenta disponibiliza uma função interessante. É a que permite detectar a ocorrência de outra ferramenta, que faça uso de técnicas semelhantes para vasculhar redes com *switch*.

Ettercap não exige a biblioteca libpcap, pois foi construído com programação direta sobre os recursos de filtragem específicos que cada plataforma oferece, no caso do linux, o *socket PF_PACKET*. Desta forma, não incorpora as facilidades de filtragem oferecidas pela biblioteca.

4.6.7 angst

Angst é um monitor de rede agressivo, que apesar de fazer uso da libpcap para capturar pacotes no meio físico, não utiliza os recursos natos de filtragem, oferecidos por ela. Diz-se que é agressivo, pois da mesma forma que o ettercap, proporciona mecanismos para capturar pacotes destinados a terceiros, em um ambiente conectado com *switch* [ANG 2001].

Além do método já apresentado em ettercap, angst faz uso de um artifício que degrada consideravelmente a performance da rede. O alvo neste caso é o *switch*. Da mesma forma que o produto chamado macof, angst inunda a rede local com números MAC randomicamente gerados. Com isso a capacidade de memória do *switch*, para armazenamento de endereços MAC se esgota e o comportamento destes fica alterado, passando a funcionar como um *hub*, enviando os pacotes para todas as portas.

Os dados capturados são registrados em um arquivo.

4.6.8 ethereal

Ethereal é um analisador de tráfego que faz uso de uma biblioteca de interface gráfica chamada GTK+. Permite a captura de pacotes por meio da libpcap e através dos recursos de interface gráfica oferecidos, torna mais fácil a tarefa de examinar o conteúdo de cada pacote. Dezenas de decodificadores de protocolos estão embutidos na ferramenta, de modo que os pacotes podem ser dissecados até os níveis mais altos [ETH 2001].

Para fazer uso do ethereal em um ambiente sem interface gráfica, existe uma alternativa, o tethereal. Este permite exibir na console o *dump* do tráfego [TET 2001].

4.6.9 etherape

Etherape é um monitor que exibe informações gráficas sobre o tráfego na rede. Fazendo uso de linhas para diferenciar comunicações entre os *hosts* e cores para identificar os protocolos utilizados, permite a visualização do fluxo de dados na rede. A biblioteca de captura utilizada é a libpcap, permitindo a obtenção dos pacotes diretamente da rede, ou por meio de um arquivo de entrada [ETA 2001].

4.6.10 tcpflow

O tcpflow é um programa que captura dados transmitidos entre as entidades de uma conexão TCP, e armazena os dados para que possam ser facilmente analisados posteriormente. Como o tcpflow entende o protocolo TCP, ele pode reconstruir a seqüência de dados de forma correta. Isto é, controla retransmissões ou chegada de pacotes desordenados [TCF 2001].

Cada fluxo de dados associados a uma conexão é armazenado em um arquivo diferente. A entrada de dados para o programa pode se dar diretamente a partir da interface de rede, ou por meio de um arquivo (*dump*) previamente gerado. Por fazer uso da libpcap, esta ferramenta permite uso de expressões de filtragem conforme o padrão utilizado por tcpdump.

4.6.11 ipfm

Ipfm é mais uma ferramenta baseada na biblioteca libpcap e é utilizada para contabilizar o volume de dados (fluxo de dados) que trafega na rede [IPF 2001].

As configurações possíveis permitem que seja selecionado um *host* específico a ser monitorado, ou podem indicar para que todo o trafego seja registrado. A saída é feita por meio de relatórios em disco, que podem ser ordenados conforme critérios específicos. O intervalo de tempo, o qual a ferramenta deverá realizar a contabilização dos dados, também é configurável.

4.6.12 ipaudit

Ipaudit é um monitor de tráfego que registra o total de tráfego gerado por cada "conexão" sobre o IP. Neste contexto, conexões correspondem a combinação de pares de endereços IP, protocolos e portas de serviço associadas [IPA 2001].

O programa pode ser alimentado diretamente na rede, mediante a captura *on-line* de pacotes, ou pela leitura de um arquivo (*dump*).

Como saída, tem-se uma linha para cada conexão detectada. As informações, que são distribuídas em colunas, reportam: IP dos *hosts* envolvidos, código do protocolo associado, portas de serviço utilizadas em ambas as entidades, volume de dados em bytes e o número de pacotes recebidos por cada entidade, e, opcionalmente, o registro de tempo correspondente à abertura e ao fechamento da conexão.

4.6.13 iptraf

Iptraf é um monitor de tráfego IP que gera algumas estatísticas, como: distribuição por tipo de protocolo, tamanho dos pacotes, portas utilizadas. A interface com o usuário é construída com ncurses [IPT 2001].

De forma semelhante ao sniffit, mantém um monitoramento dinâmico das conexões TCP ativas, mas os recursos de filtragem são escassos e apresentam problemas no momento da definição. Quanto ao monitoramento de conexões, diferentemente do sniffit não disponibiliza a visualização dos dados que trafegam, mas contabiliza o fluxo em número de bytes e de pacotes.

Com o conjunto de ferramentas acima apresentado, tem-se boas opções. Porém, é possível que para atender a uma necessidade específica, pode ser preciso fazer uso de duas ou mais ferramentas de forma combinada, ou ainda se desenvolver algo diferenciado.

4.7 A Captura na Rede

Para realizar a captura de quadros que trafegam no meio físico, é preciso fazer uso de uma biblioteca voltada a esta finalidade.

Como estudado anteriormente, onde foram descritas diferentes ferramentas que trabalham com captura de pacotes na rede, a libpcap é a biblioteca de captura utilizada por inúmeras ferramentas. Dentre estas pode-se destacar tcpdump, ethereal e snort.

Sendo que estas ferramentas citadas são amplamente utilizadas, a libpcap pode ser adotada como uma ferramenta suficiente para a captura de pacotes necessária para a construção de um monitor de rede.

4.7.1 Facilidades Oferecidas pela Libpcap

Entre outras facilidades oferecidas pela libpcap, ela permite colocar a interface de rede em modo promíscuo, possibilitando desta forma, a captura de qualquer pacote que passe pelo meio físico, independentemente de quem seja o seu destinatário.

Uma vez que todos os pacotes que passam pela rede podem ser manipulados, significa dizer que se tem uma grande quantidade de informações a serem manipuladas. A situação fica mais crítica quando o tráfego da rede está alto.

Em grande parte das situações, tem-se interesse somente sobre parte dos pacotes que estão trafegando, fazendo-se necessária então, uma seleção do tráfego.

A solução para este problema se dá pela utilização de um filtro, que é aplicado sobre os pacotes que são recebidos pela interface. Isso pode ser feito mediante o uso de comandos "if" no código do aplicativo a ser construído. Porém esta medida não é a mais adequada, sob o ponto de vista da performance. Isto é, todos os pacotes igualmente passam pelo *kernel*, sendo feita a seleção somente no nível de aplicativo do usuário.

Para melhorar a solução, é preciso que se faça a filtragem de pacotes o mais cedo possível, de forma a despender o menor tempo com pacotes que não serão selecionados [INS 2001].

A biblioteca libpcap oferece o serviço de filtragem através do *BPF (BSD Packet Filter*). Na verdade, a libpcap é a biblioteca que proporciona independência de sistema operacional ao BPF. Isto é, o BPF é o núcleo principal, e a portabilidade deste para diferentes sistemas operacionais se dá por meio da libpcap específica desenvolvida.

Falando-se sobre a funcionalidade do BPF, destaca-se [BAR 98]:

- Oferece condições para se trabalhar com diferentes interfaces, isto é, oferece uma independência do protocolo da interface;
- Proporciona um método para selecionar os dados a serem passados para a camada da aplicação.

Com o primeiro recurso, o desenvolvedor passa a ter em mãos, facilidades importantes, de modo que não precise se preocupar com questões de implementação no nível de hardware.

Já o segundo, implementa seleção de pacotes nos níveis iniciais do processamento dos mesmos. Isto é, o BPF realiza a seleção de pacotes no nível de *kernel*, repassando para o nível de usuário somente os pacotes que interessam para aplicação [McC 93].

O BPF se constitui de um conjunto de instruções de máquina. São instruções para carga a partir da memória e armazenamento na mesma, para realizar operações aritméticas e lógicas, instruções de desvio e retorno. A partir deste conjunto são montados os filtros de pacotes.

O exemplo a seguir ilustra como um filtro de pacotes para selecionar somente pacotes IP, assumindo-se estar tratando-se de uma rede Ethernet, é construído com instruções BPF:

```
        ldh     [12]
        jeq     #ETHERTYPE_IP, L1, L2
L1:     ret     #TRUE
L2:     ret     #0
```

A primeira instrução lê 4 bits da memória a partira da posição 12 do *frame* Ethernet , que corresponde ao campo tipo de protocolo. Na seqüência compara-se o valor lido com a constante que define o tipo IP. Conforme o resultado da operação lógica, um retorno diferente é dado ao filtro. No caso de falso (0), o pacote é descartado. Já no outro caso, o TRUE retornado, indica algum valor diferente de zero que representa o número de bytes a serem repassados ao próximo nível de processamento.

A funcionalidade do filtro fica mais evidente quando se trata de uma seleção mais complexa. Por exemplo, imagina-se que o filtro deva aceitar todos os pacotes IP que não foram originados em duas redes IP específicas, 128.3.112.0 ou 128.3.254.0. Se o pacote é confirmado como IP, então buscando na memória, onde está armazenado o pacote, o valor do endereço de origem. Aplica-se uma máscara de 24 bits sobre este endereço (por meio do “E” lógico) e compara-se o resultado:

```
        ldh     [12]
        jeq     #ETHERTYPE_IP, L1, L4
L1:     ld      [26]
        and     #0xffffff00
        jeq     #0x8003700, L4, L2
L2:     jeq     #0x8003fe0, L4, L3
L3:     ret     #TRUE
L4:     ret     #0
```

Obviamente o BPF possui uma sintaxe de implementação de filtros, que se aproxima da linguagem humana. As primitivas utilizadas para a definição do filtro podem ser conhecidas em [TCP 2001].

Estas primitivas são compiladas pela libpcap, e a partir delas, é montado o código que fará a seleção dos pacotes. A seguir, quando serão vistas as principais funções da libpcap, será estudada a função empregada para esta compilação.

4.7.2 Principais funções da Libpcap

Esta biblioteca é composta por mais de trinta de funções. A Tabela 4.1 a seguir apresenta as funções básicas a serem utilizadas na captura de pacotes na rede, que serão detalhadas na seqüência.

O estudo destas funções é importante para que sejam compreendidos os conceitos básicos que envolvem um processo de captura de pacotes, utilizando a libpcap. Pode-se observar as facilidades proporcionada pela biblioteca, conforme citadas anteriormente.

TABELA 4.1 - Principais Funções da Libpcap

char ***pcap_lookupdev** (char *errbuf)
int **pcap_lookupnet** (char *device, bpf_u_int32 *net, bpf_u_int32 *mask, char *errbuf)
pcap_t ***pcap_open_live** (char *device, int snaplen, int promisc, int to_ms, char *errbuf)
int **pcap_compile** (pcap_t *p, struct bpf_program *fp, char *str, int optimize, bpf_u_int32 netmask)
int **pcap_setfilter** (pcap_t *p, struct bpf_program *fp)
u_char **pcap_next** (pcap_t *p, struct pcap_pkthdr *h)
int **pcap_dispatch** (pcap_t *p, int cnt, pcap_handler callback, u_char *arg_user)
int **pcap_loop** (pcap_t *p, int cnt, pcap_handler callback, u_char *arg_user)
void **trata_pacote** (u_char *arg_user, const struct pcap_pckhdr *header, const u_char *packet)
void **pcap_close** (pcap_t *p)

O primeiro passo no processo de captura é definir a interface a ser monitorada. No caso de computadores com mais de uma interface, o usuário[2] deverá informar a interface a ser monitorada. Porém, a biblioteca possui uma função, chamada *pcap_lookupdev()*, que verifica o sistema, em busca de uma interface a ser monitorada, retornando a mesma para uma variável (*char **). Esta função possui como parâmetro uma variável do tipo *string*, de tamanho igual a *PCAP_ERRBUF_SIZE* (constante definida pela própria biblioteca), onde é armazenada a mensagem de erro caso ocorra uma falha na execução. Neste caso, o retorno da função é *NULL*.

Após definida a interface a ser monitorada, é preciso determinar o número de rede e máscara associados a este dispositivo, pois estas informações serão necessárias mais adiante, no momento da definição de filtros. O primeiro parâmetro é a interface selecionada. Na seqüência estão as variáveis que armazenam o número de rede e máscara respectivamente. Por último, a string que armazena a mensagem de erro ocorrido, caso a função retorne –1.

O próximo passo é habilitar a interface para fazer a captura de pacotes. Para isso, utiliza-se a função *pcap_open_live()*, que retorna uma sessão do tipo pcap. O primeiro parâmetro é a interface a ser monitorada. O segundo especifica o número máximo de bytes a serem capturados. Normalmente este parâmetro é definido pela constante *BUFSIZ*, que é definida em pcap.h. O terceiro parâmetro, quando verdadeiro, coloca a interface em modo promíscuo. É importante salientar que mesmo que este seja falso, não garante que a interface permaneça em modo não-promíscuo. Isso porque esta poderá ser colocada nessa situação por outro aplicativo. O quarto especifica, em milisegundos, o tempo em que estará sendo feita a captura de pacotes, é o *timeout*. Com isso é possível definir que múltiplos pacotes sejam lidos, durante um tempo determinado. Se o valor for 0, então a captura será feita até que ocorra algum erro. Caso o valor seja –1, o tempo será indefinido. Vale lembrar que em algumas plataformas o *timeout* não funciona, sendo ignorado, e a captura feita até que ocorra um erro. O último parâmetro retorna a mensagem de erro ocorrido.

[2] Usuário aqui referencia o usuário do programa, que normalmente será o *root* do sistema operacional do *host* que está fazendo o monitoramento.

Com as funções vistas até aqui, já é possível capturar pacotes na rede. Porém, quando não se está interessado em todo o tráfego, pode-se fazer uso de filtros para selecionar os pacotes a serem manipulados.

A filtragem possível através do uso da libpcap é ampla e permite uma série de combinações. É possível filtrar, por exemplo, em função: do *host* origem ou destino; de uma porta de comunicação; do protocolo... Estas são as possibilidades básicas e mais simples de serem compreendidas e implementadas. Existem também opções mais complexas, que para ser implementadas exigem maior conhecimento do mecanismo de filtragem e dos formatos do cabeçalho dos protocolos. Nesta categoria enquadram-se, por exemplo, os filtros em função dos *flags* do cabeçalho TCP, *flags* do cabeçalho IP, ou ainda, do código ICMP. Mais detalhes sobre as possibilidades e sintaxe de filtragem, podem ser adquiridos em [TCP 2001].

Para aplicar o filtro, que está armazenado em uma *string*, ao processo de captura de pacotes, é precisa antes, submetê-lo a uma "compilação". Para realizar tal operação utiliza-se *pcap_compile()*, cujo o primeiro parâmetro é a sessão obtida com *pcap_open_live()*. O parâmetro seguinte é a variável onde será armazenada a versão compilada do filtro. O terceiro é a *string* que contém o filtro definido pelo usuário. Após este, tem-se o controle de quando deve ser realizada a otimização do código de filtro resultante. Pode assumir valor 1, se verdadeiro, ou 0 se falso, sendo que nesse caso, nenhuma otimização é executada. O último parâmetro é a mascara da rede atual, que foi obtida por meio da *pcap_lookupnet()*. Caso ocorra algum problema na execução, como filtro inválido, a função retorna –1. Caso seja executada com sucesso, retornará outro valor.

Depois de compilado, o filtro precisa ser aplicado. Para isso faz-se uso de *pcap_setfilter()*. Como parâmetro tem-se a sessão e o filtro compilado retornado da função *pcap_compile()*.

Agora já é possível realizar a captura de pacotes propriamente dita. Para isso, existem duas técnicas. Uma em que capturado um pacote por vez (fazendo uso de *pcap_next()*), e outra em que o programa entra em *loop*, capturando vários pacotes conforme especificado (utilizando *pcap_dispatch() ou pcap_loop()*).

A *pcap_next ()* captura o primeiro pacote que passar pela interface selecionada, que atenda a seleção definida pelo filtro. O primeiro argumento é a sessão, na seqüência, um ponteiro para a estrutura que conterá as informações do pacote capturado. Entre as informações está o horário da captura e o tamanho do pacote. A função retorna um ponteiro para área de memória onde está armazenado o pacote capturado.

A segunda técnica é a mais útil para a confecção de um monitor. Utiliza funções do tipo *callback*. Esse tipo de função aguarda a ocorrência de um evento. Quando este ocorre, ela é executada. No caso da biblioteca pcap, ela é executada quando detecta um pacote na rede que atende as opções de filtragem.

A função *pcap_dispatch ()* é usada para capturar e processar pacotes, retornando o número de pacotes capturados. Como primeiro parâmetro esta função apresenta a sessão aberta com *pcap_open_live()*. Na seqüência, recebe o número de pacotes que devem ser processados antes que ela retorne. O terceiro argumento é o nome da função que deverá ser chamada a cada pacote detectado. O último é um ponteiro que é passado como parâmetro para a função referenciada no argumento anterior. Normalmente este último parâmetro é definido como NULL.

Quanto ao número de pacotes a serem capturados, indicam na verdade o número máximo. Pois, no caso de ter sido especificado um *timeout* em *pcap_open_live()*, quando este tempo se esgotar, a captura será concluída. Caso o valor informado para o

número de pacotes a serem capturados for negativo, pacotes serão lidos até que um *buffer* seja completado, seja esgotado o *timeout*, ou chegue ao final do arquivo (no caso de estar sendo feita uma leitura *offline).*

A outra função que usa a técnica de captura de múltiplos pacotes é *pcap_loop()*. Esta é muito semelhante a *pcap_dispatch(),* diferencia-se basicamente em função do tempo do retorno. De forma diferente, *pcap_loop()* ignora o *timeout* e não retorna até que sejam lidos o número de pacotes especificados. Caso o número informado na função seja negativo, pacotes serão capturados até que um erro ocorra, ou chegue ao final do arquivo.

Como visto acima, tanto para *pcap_loop(),* como para *pcap_dispatch(),* para cada pacote capturado é feita a chamada de uma função, que é passada por parâmetro. Esta função aqui é chamada de *trata_pacote()* e recebe três argumentos. O primeiro é o ponteiro passado no quarto parâmetro de *pcap_dispatch()* ou *pcap_loop()*. Na seqüência, tem-se o ponteiro para a estrutura com as informações sobre o pacote capturado (como em *pcap_next()*). Por último, o ponteiro para os dados do pacote capturado.

A função *pcap_close()* é utilizada ao final, para liberar os recursos da sessão.

Todo o tratamento dos dados capturados é feito dentro de *trata_pacote()*. Os dados capturados estão acessíveis por meio do ponteiro indicado no terceiro parâmetro. Porém, cabe salientar que para acessar os dados capturados, será preciso fazer uma conversão de tipos. No caso de uma rede Ethernet, será preciso converter o ponteiro u_char para um ponteiro para a estrutura de cabeçalhos de pacotes Ethernet. Depois, a partir deste endereço, conforme o protocolo do pacote capturado, será preciso fazer uso de ponteiros para estruturas de protocolos especificas.

Exemplificando, para acessar o conteúdo do cabeçalho IP, será preciso acessar o endereço que aponta para os dados capturados, incrementado do tamanho do cabeçalho Ethernet. Para acessar o cabeçalho TCP, será preciso incrementar ainda o tamanho do cabeçalho IP. A Figura 4.3 a seguir ilustra isso.

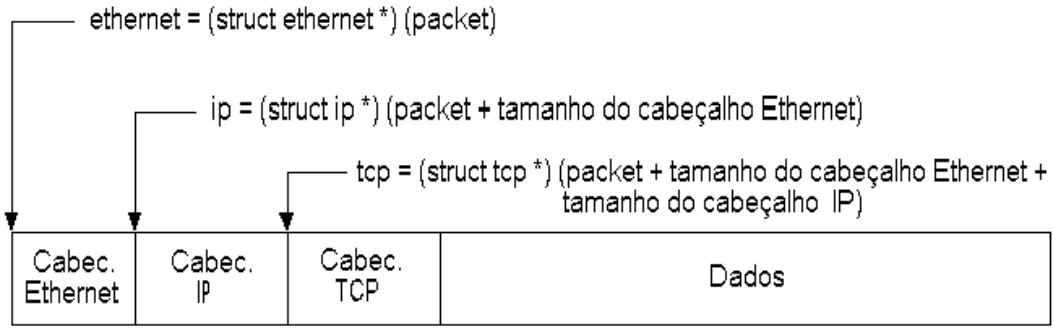

FIGURA 4.3 - Acessando o pacote memória.

Os tipos de estruturas normalmente são definidas em /include/netinet. Entre estas, pode-se citar estruturas para o cabeçalho Ethernet, para o cabeçalho IP, ou TCP, por exemplo.

A partir deste ponto, pode-se manipular os dados dos cabeçalhos de cada um dos protocolos.

5 O Monitor de Transações Internet (MONITIN)

O objetivo deste estudo é buscar formas de se construir um monitor de transações que ocorrem na rede sobre o protocolo internet. Este monitor deve ser capaz de fornecer informações sobre conexões ativas e sobre o fluxo de pacotes entre as entidades da rede.

Foi feito o estudo para definir como implementar uma ferramenta deste tipo e a construção de um protótipo para provação.

A ferramenta proposta, de agora em diante será denominada MONITIN (MONItor de Transações INternet).

A construção do monitin passou pelas seguintes etapas:

- Definição da estrutura de hardware necessária;
- Definição dos recursos de software exigidos;
- Construção do algoritmo de monitoramento das conexões;
- Construção do protótipo para provação.

5.1 Definição da estrutura de hardware necessária

O monitor deve operar sobre o padrão de rede Ethernet. A escolha do Ethernet se deve ao fato desta tecnologia ser amplamente difundida e utilizada [SPU 2000].

Conforme estudado no capítulo 4, para se fazer o monitoramento de redes Ethernet tem-se duas alternativas básicas. Por passagem ou por meio de um monitoramento com a interface em modo promíscuo.

O monitoramento em modo promíscuo é melhor para esta situação porque:

- Permite que se analise toda a comunicação que ocorre dentro de um domínio de colisão;
- É transparente aos usuários e aplicações da rede. Isto é, como o monitoramento se dá em modo passivo, ele pode ser desativado e ativado a qualquer momento sem interferir no funcionamento normal da rede;
- Possibilita que o monitoramento seja realizado em pontos diferentes dentro do domínio de colisão;
- Não existe a formação de gargalo para o fluxo do tráfego.

Por outro lado, o monitoramento passivo permite que alguns pacotes passem pelo monitor sem serem analisados. Isso tende a ocorrer com maior freqüência quando o tráfego é mais intenso.

Mas considerando o propósito da ferramenta em questão, entende-se que perfeitamente possível aceitar este tipo de situação, sem comprometer a funcionalidade da mesma. Pesa também nessa decisão, que a idéia principal é se ter uma ferramenta que seja independente aos demais usuários da rede. Pois se a analise de todos os pacotes fosse uma exigência obrigatória, sem admitir qualquer perda, seria uma exigência o uso de um monitoramento por passagem de pacotes, mas para isso o monitor perderia flexibilidade.

5.2 Definição dos recursos de software exigidos

A decisão do software neste contexto, diz respeito ao sistema operacional. Nesta questão, a família Linux é indicada por uma série de fatores:

- trata-se de um sistema de software livre, e por isso, traz uma série de vantagens implícitas, como possibilidade de alteração, cópia ou distribuição [FSF 02];
- o custo do Linux é baixo comparado com outros sistemas operacionais proprietários;
- existe um grande número de desenvolvedores para Linux na comunidade de desenvolvimento mundial, e desta forma, existe um grande número de bibliotecas disponíveis para os diferentes propósitos;
- o Linux está em ascendência em nível mundial. Isto é, o número de computadores com Linux instalado está em crescimento;

Entre as diversas distribuições de Linux existentes, a escolhida para os testes e construção do protótipo é a RedHat. Isso se dá pois esta distribuição é muito difundida mundialmente, sendo homologada como compatível com uma série de outros *softwares* e banco de dados amplamente utilizados (como produtos Netware e Oracle) [NOV 2001] [ORA 2001]. Este é um indício de que é reconhecida como uma distribuição de qualidade.

Para reforçar a escolha, o RedHat é a distribuição predominante na instituição onde será implementado o protótipo e feitos os testes de uso do monitin.

5.3 Construção do algoritmo de monitoramento das conexões

Para fazer o monitoramento do tráfego na rede Ethernet, a primeira providência a ser tomada é a forma como será feito o acesso aos pacotes que estão trafegando no meio físico.

Como o propósito deste estudo é construir um monitor passivo, então será preciso primeiramente colocar a interface de rede que se quer monitorar, em operação, no modo promíscuo.

O passo seguinte é obter acesso aos pacotes que estão trafegando e selecionar os que serão repassados para a aplicação, para que esta possa manipulá-los.

Para cada pacote recebido pela aplicação (pacote que foi selecionado previamente) deve ser executado o procedimento de depuração, fazendo a demultiplexação em nível de camada internet [COM 98]. Tal demultiplexação consiste em classificar os pacotes quanto aos protocolos de níveis mais altos, isto é, UDP, TCP, ICMP...

Feito o enquadramento do pacote quanto ao protocolo embutido no IP, tem-se então acesso aos campos específicos dos cabeçalhos dos protocolos em questão.

Grande parte das aplicações IP existentes, são construídas sobre TCP. Isso se deve ao fato deste oferecer serviço voltado a conexões, o que não ocorre com o UDP. Portanto, quando se fala em monitorar as conexões sobre o IP, monitora-se os pacotes do tipo TCP que rodam sobre o IP.

Como o monitin deve operar de forma *on-line*, podendo ser iniciado a qualquer momento, então deverá identificar a abertura de conexões, as conexões em andamento, bem como a troca de pacotes que indica o encerramento das mesmas.

O TCP possui uma máquina de estados que define a situação das conexões. Porém, nesta implementação, o algoritmo de controle de conexões não segue fielmente esta seqüência de estados. Como pode ser visto a seguir, o algoritmo de controle de conexões é simplificado. A simplificação do algoritmo é importante para que se reduza o tempo de processamento de cada pacote capturado.

Com a simplificação do algoritmo algumas situações reais da máquina de estados do TCP podem não serem detectadas. Porém, para a finalidade desta ferramenta, estes casos são irrelevantes, não comprometendo a utilidade da mesma.

Nesta visão simplificada, tratar as conexões TCP consiste basicamente na análise de 3 *flags*, contidos no cabeçalho destes pacotes:

- ACK, para a detecção de um sentido da conexão;
- FIN, para o encerramento de um sentido da conexão;
- RST, para o encerramento de uma conexão em ambos os sentidos;

Para o reconhecimento de uma conexão, o monitin não pode deter-se ao reconhecimento do *handshake* de três vias, que ocorre no momento do estabelecimento de uma conexão. Se trabalhasse desta forma, o monitor não reconheceria as conexões em andamento.

Para isso, basta que se identifique os pacotes TCP que possuem o *flag* ACK ativado. Uma vez que um ACK significa que uma entidade está informando a outra sobre o recebimento de um pacote anteriormente enviado, configura-se a existência de uma conexão.

Quando um pacote ACK é identificado, então a informação da conexão no sentido de tráfego do mesmo é armazenada na lista de conexões existentes. Se já existir uma entrada nesta lista para a conexão identificada, então os dados referentes serão atualizados, caso contrário, será aberta uma nova entrada na lista.

Desta forma, para cada conexão se terá duas entradas na lista. Uma para cada sentido do tráfego. Isto se faz necessário para que seja possível monitorar o sentido do fluxo existente em cada conexão ativa.

Parece funcionar, mas fazer simplesmente a análise do ACK não é eficiente. O problema surge com o ACK originado como resposta de uma aceitação de um FIN. Da forma proposta até o momento, este ACK seria interpretado como uma conexão em andamento, sendo que na verdade ele é um sinal de confirmação do fechamento de uma conexão anterior.

Para solucionar este caso, poderia-se manter um controle de estados de cada conexão, mas isso complicaria o algoritmo e geraria um aumento na carga de processamento. A melhor solução encontrada, foi ignorar os pacotes ACK que não estejam transportando dados. Esta medida é eficiente, pois o pacote que contém o último ACK, que confirma o fechamento do último sentido da conexão, não deve transportar dado nenhum, simplesmente ser um sinalizador de confirmação de recebimento.

Por ventura alguns ACK serão ignorados durante o tempo de conexão, porém, esta medida não prejudica a funcionalidade do monitor, uma vez que se nenhum dado está contido no pacote. Desta forma, não influencia na contabilização dos dados referentes ao volume de fluxo de informações entre as entidades.

Quando um FIN é detectado, é feita a busca da conexão na lista das conexões ativas. Caso seja encontrada, a conexão no sentido de tráfego do pacote contendo o FIN, é retirada da lista. Caso não seja encontrada, o pacote é ignorado.

Quando um RST é detectado, ambos os sentidos da conexão são localizados e eliminados da lista de conexões ativas, sendo que nada é feito quando as respectivas entradas na lista não são encontradas.

Com estas operações sobre uma lista dinâmica que se mantém ordenada em função do endereço IP e porta TCP da origem, tem-se o controle das conexões ativas. A ordenação da lista é feita não somente para a melhor visualização no momento da exibição, mas para otimizar o mecanismo de procura dentro da lista.

O passo seguinte é a exibição das conexões existentes. Para isso a lista é varrida por completo.

A Figura 5.1 abaixo ilustra o algoritmo utilizado. Pode-se observar que após a configuração da interface e a definição do filtro, o processo entra em *loop*, tratando os pacotes selecionados, até que um erro ocorra.

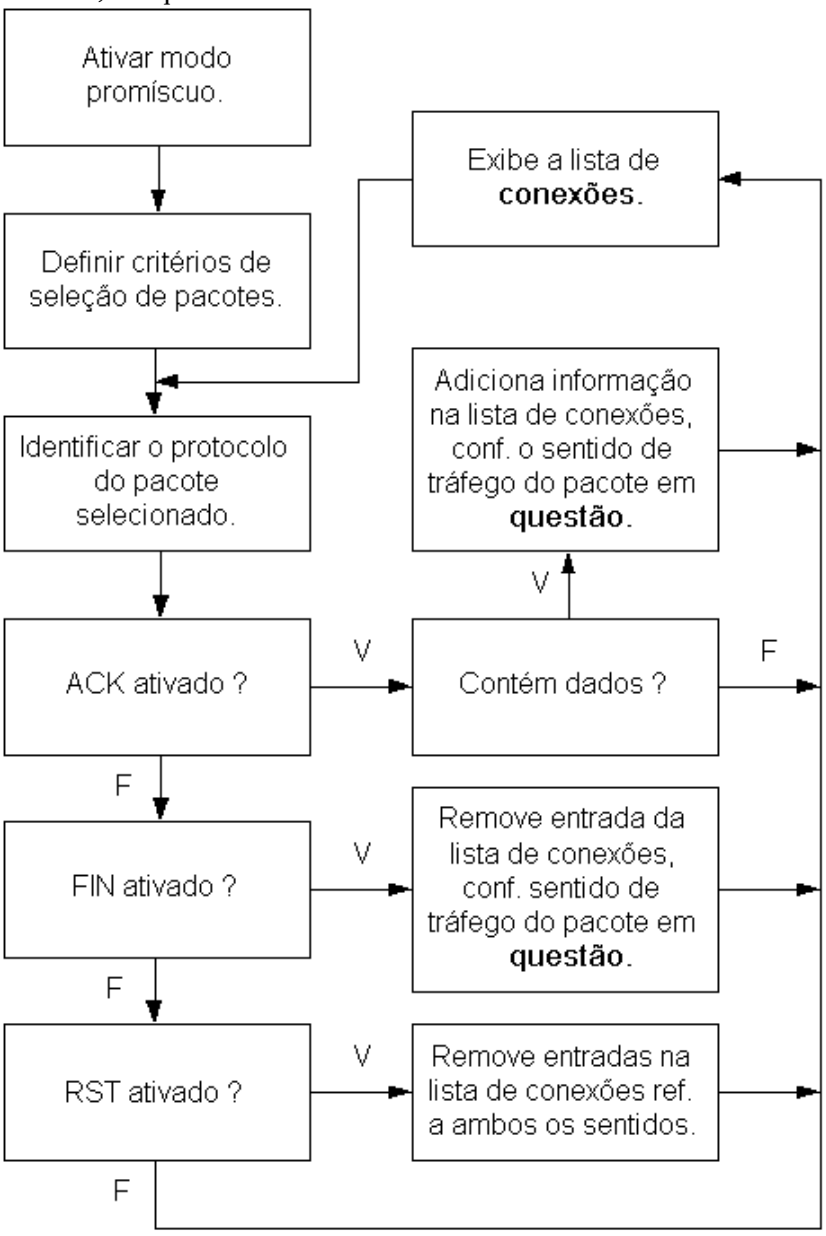

FIGURA 5.1 - Algoritmo utilizado para monitorar conexões

Com esta estrutura de manipulação de pacotes, é possível se implementar outras funcionalidades no monitin, além da originalmente proposta, que foi vista acima.

O monitin pode monitorar o volume de tráfego entre duas entidades quaisquer, independentemente do protocolo que está sendo transportado sobre o IP. Para isto, basta que para cada pacote selecionado gere uma manutenção em uma lista dinâmica. Esta lista contém informações como o endereço IP do par de computadores envolvidos,

trafego e número de pacotes contabilizados. A ordenação pode ser feita pela máquina origem ou pela máquina destino, dependendo de uma escolha previamente feita.

Outra funcionalidade interessante de se implementar é a exibição dos conteúdos dos campos que constituem o cabeçalho de um pacote. Para isso é preciso que para cada pacote selecionado, sejam exibidos os dados previamente definidos.

Estas características foram implementadas no protótipo que está descrito abaixo.

5.4 Construção do protótipo para provação

O protótipo do monitin foi construído em linguagem C e encontra-se em situação operacional, sendo possível a sua utilização.

Para realizar a tarefa de captura de pacotes no meio físico, o monitin faz uso da libpcap. Como já mencionado anteriormente, esta biblioteca proporciona a independência de sistema operacional e abstrai detalhes de implementação em nível de *hardware*, quanto ao tipo de interface de rede a ser utilizada. A libpcap oferece ainda condições de se fazer a filtragem de pacotes no *kernel*. Desta forma, o processamento será otimizado, pois somente chegarão ao nível da aplicação os pacotes selecionados pelo usuário da ferramenta.

Em outras palavras, com o uso dos recursos do filtro de pacotes BPF, embutido na libpcap, pode-se fazer a seleção dos pacotes que chegam ao monitin para serem processados, descartando os que não fazem parte do contexto a ser monitorado.

A interface de rede deve trabalhar em modo promíscuo, a fim de aceitar todos os pacotes que passam pelo meio físico, mesmo que não sejam a ela endereçados. Com isso, conclui-se que a quantidade de dados a ser manipulada é grande. Quanto maior for o tráfego na rede, maior será o volume a ser processado. Por conseqüência, maior é o risco de esgotar os recursos do computador que está executando o monitor e, com isso, deixar pacotes que trafegam pela rede passarem despercebidos.

Em função deste problema, torna-se mais valorizado o serviço de filtro de pacotes oferecido pela libpcap. Um filtro bem definido aprimora a eficiência do monitor, conforme o enfoque de monitoramento desejado.

O usuário do monitin pode definir a seleção de pacotes a ser aplicada, mediante a montagem de uma expressão de filtragem. Esta expressão é passada ao monitor via parâmetro da linha de comando, no momento da chamada da ferramenta.

Também é passada por parâmetro a opção que define a escolha da funcionalidade do monitin, descrita na seção anterior, que será ativada.

As opções de chamada são as seguintes:

```
monitin    { [ filtro] }      [ opções ]
```

onde:

`filtro` = expressão de seleção de pacotes, utilizando as primitivas e a sintaxe do BPF; isto é, o mesmo formato utilizado para definir os filtros do tcpdump. Esta expressão de seleção define quais pacotes o *kernel* repassará para o monitin. Caso nenhum filtro seja definido, assume-se que todos os pacotes devem ser analisados. Maiores detalhes quanto a sintaxe, em [TCP 2001].

opções = são as chaves que ativam diferentes características de trabalho ao monitin. São elas:

[-c | -d | -td | -to | -x]

Cada opção deve ser utilizada separadamente, sendo que a opção *default*, que é ativada mesmo quando nenhuma opção é definida no momento da chamado do monitin é a "-x".

A seguir, cada opção é detalhada, explicando a função exercida pelo monitin, quando associada a ele.

```
...
Pacotes capturados = 211  IP = 211   NIP = 0
Pacotes capturados = 212  IP = 212   NIP = 0
Pacotes capturados = 213  IP = 213   NIP = 0
Pacotes capturados = 214  IP = 214   NIP = 0
Pacotes capturados = 215  IP = 215   NIP = 0
Pacotes capturados = 216  IP = 216   NIP = 0
Pacotes capturados = 217  IP = 217   NIP = 0
Pacotes capturados = 218  IP = 218   NIP = 0
Pacotes capturados = 219  IP = 219   NIP = 0
...
```

FIGURA 5.2 - Exibição do contador de pacotes

-c : funciona somente como contador de pacotes. Nenhum tratamento especial é dado ao pacote, somente são contabilizados os pacotes que são tratados pelo monitin, classificando-os em pacotes IP ou não IP. A saída é feita para cada pacote capturado. A Figura 5.2 ilustra a saída do contador de pacotes.

-d : esta opção faz uso de um arquivo auxiliar chamado "param.txt" que deve estar no mesmo diretório do monitin. Com esta combinação é possível se visualizar o conteúdo de diferentes campos IP e TCP de um pacote, com exceção do campo de dados. Quando esta opção é utilizada, o arquivo param.txt é analisado, e os campos definidos como 1 (verdadeiro) serão mostrados. A Figura 5.6 representa este arquivo, enquanto a Figura 5.3 reflete a execução onde é solicitado para que sejam exibidos:

- o tamanho do pacote;
- os endereços e portas origem e destino;
- o tamanho da janela.

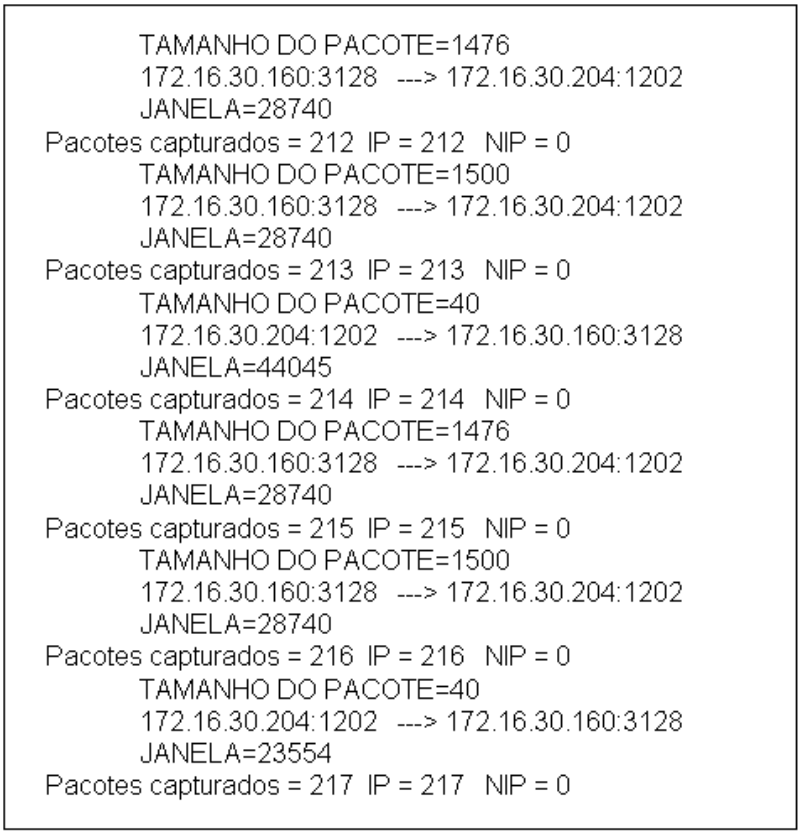

FIGURA 5.3 - Exibição dos campos TCP/IP selecionados

`-td`: exibe o fluxo de tráfego entre as diferentes entidades IP, ordenado pelo endereço destino. Com esta opção é visualizado também o número de pacotes trocados entre cada par de entidades. A Figura 5.4 representa a saída da execução do monitin, para contabilização do tráfego, ordenado pelo endereço IP destino.

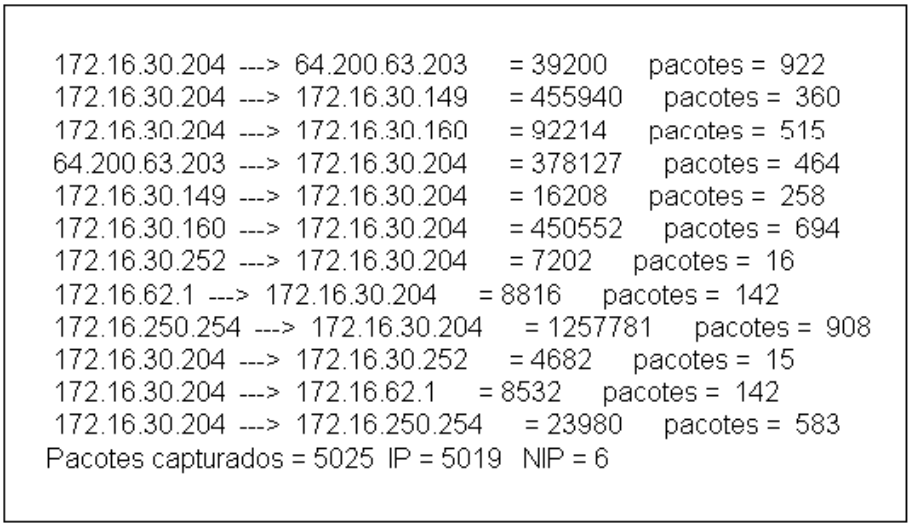

FIGURA 5.4 - Contabilização do tráfego, ordenada pelo IP destino.

`-to`: similar a opção **`-td`**, porém é ordenada em função da entidade origem.

`-x` : (opção *default*) exibe as conexões sobre o protocolo internet ativas. Como o TCP é o protocolo da família TCP/IP que é voltado a conexões, na verdade aqui são visualizadas as conexões TCP. A Figura 5.5 mostra como é feita a visualização das conexões monitoradas.

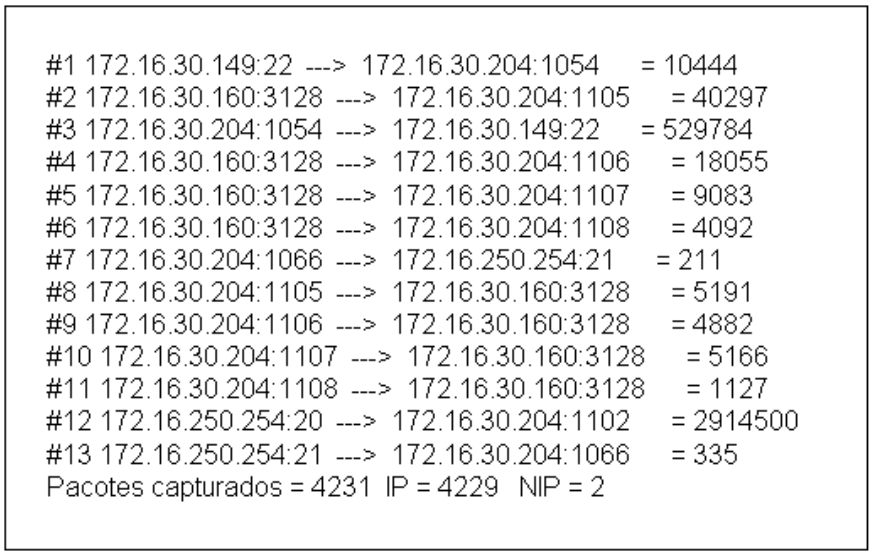

FIGURA 5.5 - Visualização das conexões ativas

O arquivo “param.txt” precisa ser alterado pelo usuário a cada vez que este quiser mudar os campos TCP ou IP a serem visualizados. O arquivo possui um formato padrão que não pode ser mudado pelo usuário. Caso este arquivo seja alterado na sua

estrutura, por exemplo, por meio da exclusão de uma linha, o funcionamento do monitin para a opção "**-d**" estará comprometido. Tendo em vista possíveis problemas que possam ocorrer na edição deste arquivo, foi criado um programa chamado "gera_param". Este programa tem a única e exclusiva finalidade de gerar o arquivo "param.txt" inicial.

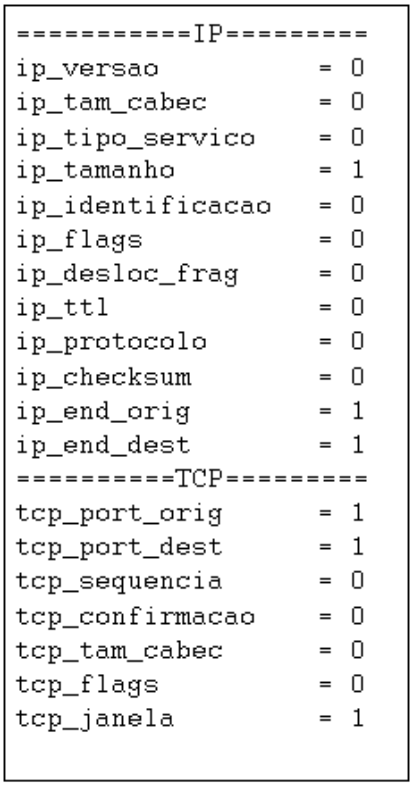

FIGURA 5.6 - Formato do Arquivo Param.txt

A configuração inicial deste arquivo é dada pelo formato exibido na Figura 5.6. A partir deste formato, podem ser feitas as alterações. Os campos que o usuário deseja visualizar são definidos como 1 (um) e os demais como 0 (zero).

No formato inicial criado pelo "gera_param", são visualizados os endereços IP de origem e destino dos pacotes, juntamente com as respectivas portas de serviço utilizadas, o tamanho do pacote IP e o tamanho da janela TCP.

6 Validação do Protótipo

Após a construção do protótipo, baseado no algoritmo e nas definições feitas, é preciso realizar testes para validar o funcionamento do monitin.

Para os testes foram utilizados 3 computadores denominados Comp1, Comp2 e Comp3, com as características descritas na Tabela 6.1.

TABELA 6.1 - Descrição dos computadores utilizados para os testes

Nome	**Comp1**	**Comp2**	**Comp3**
Processador	Pentium II–350 Mhz	Pentium II–350 Mhz	Pentium III-1Ghz
Memória	64 mb	128 mb	512 mb
Sistema Operacional	RedHat 7.2	RedHat 7.2	RedHat 7.2
Kernel	2.4.7-10	2.4.7-10	2.4.7-10

Os testes foram executados em uma rede Ethernet de 100 Mbps, onde além dos computadores descritos acima, estão conectados diretamente, no mesmo domínio de colisão, alguns servidores. Esta estrutura física do ambiente de testes está ilustrada na Figura 6.1.

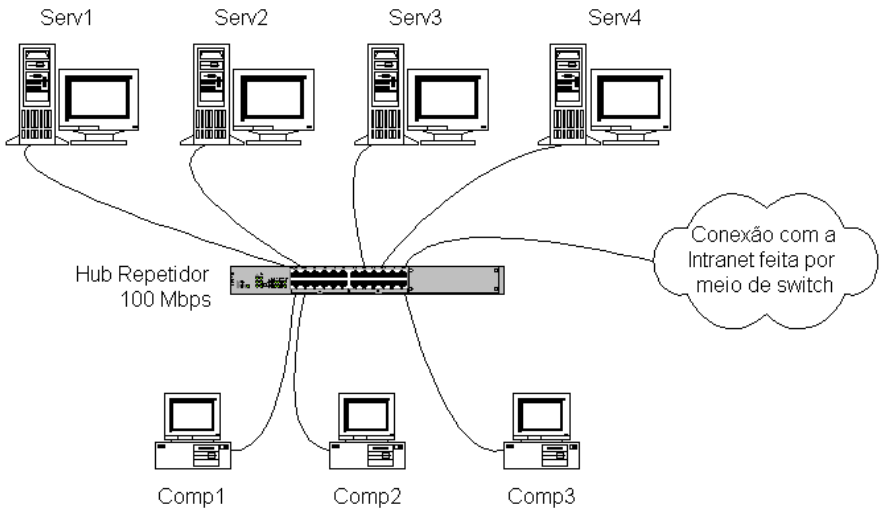

FIGURA 6.1 - Ambiente de testes

A validação do funcionamento do monitin foi feita em duas fases distintas:

- A primeira enfoca a contagem de pacotes capturados na rede, a fim de apurar a perda de pacotes;
- A segunda faz o monitoramento de conexões ativas, com a finalidade de validar o algoritmo utilizado.

6.1 Testando a captura de pacotes

Esta operação objetiva testar a capacidade do monitin em capturar pacotes que trafegam na rede. São realizadas seis medições de captura, em tempo diferente. Para cada medição, ocorre variação do tráfego existente na rede, isto é, muda-se a carga de rede. No anexo 1, estão as figuras que representam o comportamento do tráfego durante cada uma das medições feitas.

A variação na carga da rede é provocada através da criação de tráfego entre os servidores conectados ao domínio de colisão.

Em cada medição, cada um dos computadores Comp1, Comp2 e Comp3 executam o monitin juntamente com o tcpdump, com as mesmas opções de filtragem, a fim de se comparar a quantidade de pacotes capturados por cada uma das ferramentas e as diferenças no comportamento destas, em cada computador.

Em cada computador os seguintes comandos são executados simultaneamente (em terminais diferentes):

a) monitin { } –c
b) tcpdump –n –nn
c) monitin { ‘src 172.16.30.160 or dst 172.16.30.104’ } -c
d) tcpdump -n –nn ‘src 172.16.30.160 or dst 172.16.30.104’
e) monitin { dst 172.16.30.104 } -c
f) tcpdump –n –nn dst 172.16.30.104

Os primeiros dois comandos capturam todos os pacotes que estão passando pelo meio físico e será chamado de filtro1. O terceiro e quarto fazem a seleção dos pacotes cuja origem seja 172.16.30.160 ou o destino seja 172.16.30.104, sendo chamado de filtro2. Já o quinto e sexto comando, capturam somente pacotes cujo destino é 172.16.30.104, aqui denominado de filtro3.

Os parâmetros –n e –nn no tcpdump são utilizados para que o mesmo não converta portas e endereços para nomes, tornando-se desta forma, mais rápido.

O tcpdump foi utilizado como referência de verificação, pois trata-se de uma ferramenta popular para coleta de tráfego [MER 2000], além de ser freqüentemente utilizada como ponto de aferição para avaliar outras ferramentas [TAY 2001] [SMI 2001]. É sabido que o tcpdump, como outras ferramenta de captura de tráfego, pode perder pacotes. Porém, como é uma ferramenta muito utilizada nesta área, será utilizado como base comparativa. Mesmo que ele não possa oferecer uma confiabilidade total quanto ao número de pacotes perdidos, o valor referente ao número de pacotes capturados com sucesso servirá de parâmetro para avaliar o monitin.

Para que as diferentes estações de monitoramento (Comp1, Comp2 e Comp3) iniciem a contabilização de pacotes ao mesmo tempo, o *hub* ao qual estão conectadas é desligado enquanto são preparadas. Após todos as ferramentas estarem prontas para iniciar a contagem de pacotes, o *hub* é ligado, ativando o tráfego no domínio de colisão. O mesmo procedimento de desligamento do *hub* é realizado para encerrar simultaneamente a captura de pacotes.

TABELA 6.2 - Número de pacote capturados nas medições com Filtro1

Filtro1 = todos os pacotes						
	Comp1		**Comp2**		**Comp3**	
	Monitin	**Tcpdump**	**Monitin**	**Tcpdump**	**Monitin**	**Tcpdump**
Medição 1	2.711.034	2.885.783	2.698.487	2.885.783	2.885.783	2.885.783
Medição 2	4.013.291	4.450.803	4.017.094	4.450.803	4.371.140	4.450.794
Medição 3	5.702.031	6.640.078	5.705.258	6.640.078	6.486.899	6.640.065
Medição 4	5.815.560	6.760.509	5.827.470	6.760.504	6.606.543	6.760.500
Medição 5	6.310.737	7.916.192	6.318.089	7.916.192	7.631.254	7.916.186
Medição 6	7.461.321	10.001.902	7.469.491	10.001.901	9.996.370	10.002.100

A Tabela 6.2 representa os pacotes contabilizados com o filtro1. Enquanto as medições feitas com filtro2 e filtro3 estão descritas na Tabela 6.3 e Tabela 6.4, respectivamente.

TABELA 6.3 - Número de pacote capturados nas medições com Filtro2

Filtro2 = 'src 172.16.30.160 or dst 172.16.30.104'						
	Comp1		**Comp2**		**Comp3**	
	Monitin	**Tcpdump**	**Monitin**	**Tcpdump**	**Monitin**	**Tcpdump**
Medição 1	74.488	74.488	74.488	74.488	74.488	74.488
Medição 2	74.488	74.488	74.488	74.488	74.488	74.488
Medição 3	42.295	42.295	42.295	42.295	42.295	42.295
Medição 4	61.316	61.316	61.316	61.316	61.316	61.316
Medição 5	44.118	44.118	44.118	44.118	44.118	44.118
Medição 6	355.764	355.764	355.764	355.764	355.764	355.764

No anexo 1, pode-se visualizar os gráficos que representam o tráfego existente no momento de cada medição. Nas figuras, pode-se observar que a curva de tráfego inicia e termina em zero. Isso se deve ao fato de o *hub* ter sido desativado antes e depois de cada medição.

TABELA 6.4 - Número de pacote capturados nas medições com Filtro3

Filtro3 = dst 172.16.30.104						
	Comp1		**Comp2**		**Comp3**	
	Monitin	**Tcpdump**	**Monitin**	**Tcpdump**	**Monitin**	**Tcpdump**
Medição 1	25.993	25.993	25.993	25.993	25.993	25.993
Medição 2	32.443	32.443	32.443	32.443	32.443	32.443
Medição 3	27.079	27.079	27.079	27.079	27.079	27.079
Medição 4	29.510	29.510	29.510	29.510	29.510	29.510
Medição 5	33.682	33.682	33.682	33.682	33.682	33.682
Medição 6	31.294	31.294	31.294	31.294	31.294	31.294

Analisando os dados coletados, observa-se que para as situações em que se possui um filtro que seleciona uma pequena quantidade de pacotes em relação ao

tráfego total, dando um enfoque ao monitoramento (filtro2 e filtro3), as duas ferramentas se comportam igualmente, nos três computadores testados.

No filtro1, onde os dados não são previamente selecionados pela libpcap, ocorre variação no número de pacotes capturados. Sem a aplicação de filtro algum, o monitin tem um comportamento menos eficiente que o tcpdump, em todas as situações, excetuando a amostragem 1, no computador denominado Comp3. Nesta situação, tem-se um tráfego baixo e um processador de alto desempenho. Com estes dados fica evidente que o processador tem grande importância no desempenho desta ferramenta, uma vez que mesmo com tráfego inferior a 10 Mbps, as medidas de monitin sobre Comp1 e Comp2 não se equivaleram às obtidas com o tcpdump.

Ainda analisando o filtro1, pode observar que para cada medição, mesmo em computadores diferentes, o tcpdump manteve-se mais estável do que o monitin. Em outras palavras, o número de pacotes capturado pelo tcpdump em cada computador (Comp1, Comp2 e Comp3) não possui grande variação. Esta constatação pode ser visualizada na Figura 6.2.

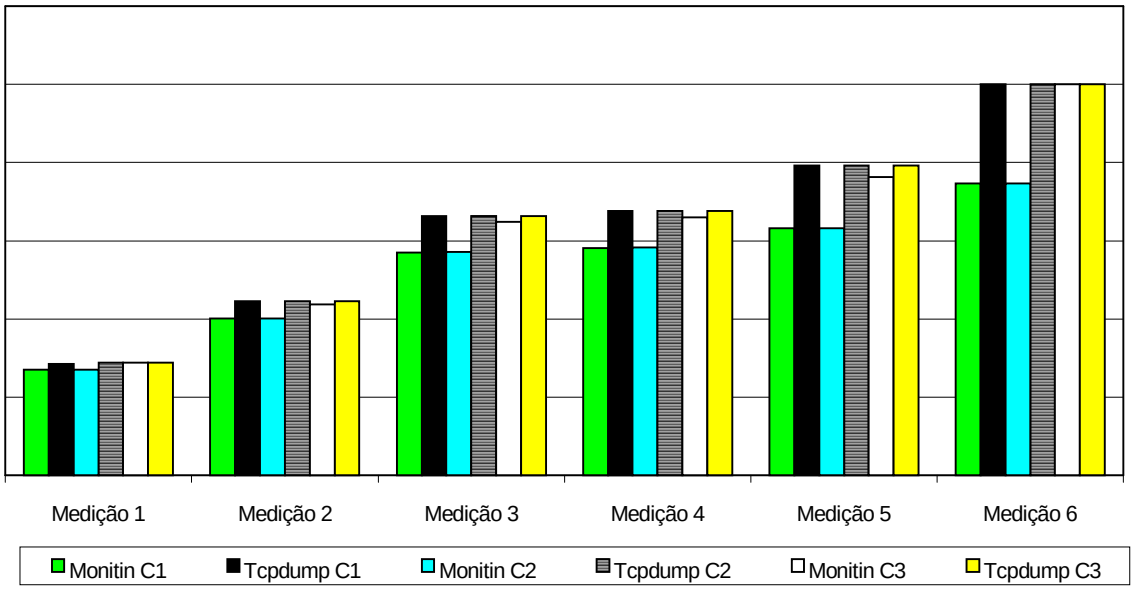

FIGURA 6.2 - Pacotes capturados em cada medição, com filtro1

A Figura 6.3 ilustra a diferença entre monitin e tcpdump na captura de pacotes, no mesmo computador, quando nenhuma seleção de pacotes é feita, isto é, quando filtro1 foi aplicado. Nesta ilustração pode-se observar que mesmo Comp2 tendo mais memória, não obteve grande vantagem sobre Comp1. Já Comp3, que possui maior capacidade de processador, teve grande ganho, e a diferença entre os dados obtidos pelo monitin e o tcpdump neste computador, não passou de 3,6 %. Para melhor compreensão da figura, é importante salientar que as primeiras medições foram feitas com os picos máximos de tráfego estando abaixo dos 10 mbps. Nas medições seguintes, o tráfego foi sendo incrementado, sendo que na quinta e sexta medição, atingiu os maiores picos, ver anexo 1.

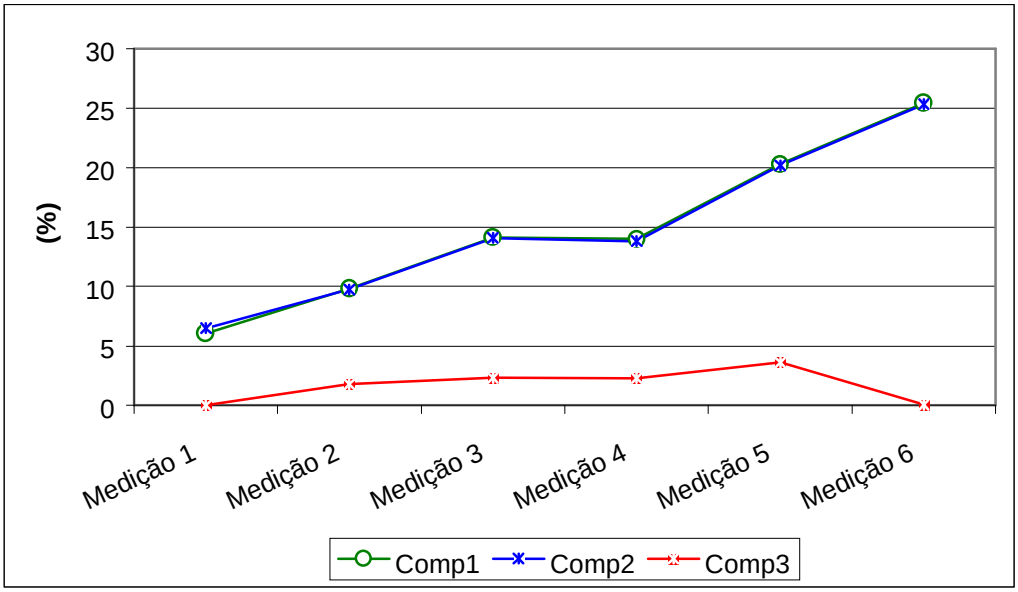

FIGURA 6.3 - Diferença de pacotes capturados entre Monitin e Tcpdump com filtro1

Os dados obtidos com filtro2 e com filtro3 mostram a equivalência entre o número de pacotes capturados pelo monitin e pelo tcpdump, em cada um dos computadores, em todas as medições feitas. Isso revela que para situações em que o enfoque do monitoramento é reduzido, e eficiência do monitin é semelhante a do tcpdump, diferentemente da situação que ocorre quando todo o tráfego deve ser analisado. Isto se deve à filtragem feita em nível de *kernel* com os recursos da libpcap. Com esta filtragem, nem todos os pacotes que atingem a interface de rede são repassados para a aplicação final. É feita uma seleção prévia, conforme a expressão de filtragem definida, fazendo com que a ferramenta tenha que tratar um número menor de informações.

6.2 Validando o algoritmo de monitoramento de conexões

A validação do algoritmo utilizado para fazer o monitoramento das conexões é realizada em duas etapas, sempre fazendo um comparativo entre os resultados obtidos a partir do monitin, com os obtidos com outra ferramenta de monitoramento de conexões. A ferramenta comparada é o sniffit.

O sniffit foi escolhido por ser uma ferramenta que opera, de forma semelhante ao monitin, sem necessidade do ambiente gráfico, e monitora a rede de forma passiva. Além disso, está já é a ferramenta adotada para a tarefa de monitoramento de conexões, no ambiente institucional em que estão sendo realizados os testes.

6.2.1 Primeira etapa

Primeiramente foi gerado um *dump* com o tcpdump, a partir do monitoramento de um *host* especifico, identificado pelo número 172.16.30.104.

Durante a geração do *dump*, no computador alvo, foram abertas várias conexões web, realizado FTP de vários arquivos menores de 500 kbytes, sendo ainda realizada

uma conexão de áudio *online* com um servidor. Chegou-se a atingir um pico de 28 conexões simultâneas. Quando todas as conexões foram encerradas, o *dump* foi concluído.

Utilizando-se uma ferramenta chamada tcpreplay, o tráfego armazenado foi posteriormente injetado novamente na rede, em situações diferenciadas de carga de rede.

Os gráficos que demonstram o tráfego de rede presente no momento da injeção do "tráfego conhecido" estão no anexo 2, estando o intervalo de tempo em que ocorreu a injeção de pacotes, limitado entre as linhas verticais.

Antes de se fazer a injeção do tráfego, os três computadores responsáveis pelo monitoramento foram preparados, para monitorar o *host* alvo, com a seguinte linha de comando:

```
monitin { host 172.16.30.104 } -x
```

Foram também preparados para fazer o monitoramento como sniffit.

É importante mencionar que antes de se fazer a injeção do tráfego, as entidades envolvidas no tráfego previamente capturado, foram desativadas, a fim não se perder o controle da situação referente a conexões ativas, durante a validação.

Os dados obtidos, nas quatro amostragens feitas, são exibidos na Tabela 6.5.

As linhas da Tabela 6.5 representam cada um dos ciclos de injeção e coleta de informações. As colunas são divididas primeiramente em função dos computadores onde foram executados os monitores (Comp1, Comp2 e Comp3).

Para cada computador, existe uma subdivisão em 2 colunas, uma para cada ferramenta de monitoramento utilizada (monitin e sniffit). Para o monitin ainda é feita mais uma divisão em colunas. Uma indica o número de conexões restantes ao final do monitoramento e outra, o número de pacotes capturados. Para o sniffit, somente é registrada a informação de conexões restantes no final do período.

TABELA 6.5 - Medições do monitoramento de conexões - etapa 1

	Comp1			**Comp2**			**Comp3**		
	Monitin		**Sniffit**	**Monitin**		**Sniffit**	**Monitin**		**Sniffit**
	Conex	**Pacot**	**Conex**	**Conex**	**Pacot**	**Conex**	**Conex**	**Pacot**	**Conex**
Medição 7	0	44.216	18	0	44.216	18	0	44.216	18
Medição 8	0	44.216	19	0	44.216	20	0	44.216	18
Medição 9	0	44.215	18	0	44.216	18	0	44.216	18
Medição 10	0	44.215	19	0	44.214	20	0	44.214	19

Como o *dump* foi concluído quando todas as conexões já haviam sido encerradas, é de se esperar que ao final da injeção, os monitores não apresentem nenhuma conexão ativa.

Analisando a tabela de resultados, observa-se que o monitin comportou-se de forma satisfatória, enquanto o sniffit acusou ao final, a existência de conexões ativas, o que não é real.

Pode-se constatar também que em momentos de tráfego mais elevado (amostragens 9 e 10), o monitin acusou perda de pacotes, mas não comprometeu o resultado final.

6.2.2 Segunda etapa

O procedimento adotado aqui é semelhante ao anterior.

Foi gerado um *dump*, a partir do monitoramento de um host especifico, identificado pelo número 172.16.30.104.

Desta vez, o objetivo foi atingir um pico de conexões simultâneas mais alto. Tal situação serviu para verificar o comportamento do monitin, quanto existe um número maior de conexões para controlar. O pico atingido chegou a 45 conexões, e o processo de *dump* foi encerrado quando ainda existiam 6 conexões ativas.

Numa primeira tentativa de obtenção de resultados, o sniffit apresentou como resultado final, mais de uma centena de conexões simultâneas, o que de fato não existiu. Este fato caracteriza problemas no reconhecimento do encerramento de conexões por parte desta ferramenta.

Como não opera de forma satisfatória, o sniffit foi retirado dos testes comparativos e a os valores obtidos com ele, foram ignorados. Se ele não elimina as conexões que são fechadas, tende a acumular inúmeras conexões simultâneas, exigindo muito recurso processamento e memória, deixando o computador lento, podendo prejudicar os testes do monitin.

O tráfego capturado equivale a 20 minutos de comunicação, onde foram capturados 326.784 pacotes, e as três injeções foram feitas sobre um tráfego médio de 20 Mbps, conforma pode ser visualizado no anexo 3.

TABELA 6.6 - Medições do monitoramento de conexões - etapa 2

	Comp1		**Comp2**		**Comp3**		**Número Médio de Pacotes**
	Monitin		**Monitin**		**Monitin**		
	Conex	**Pacot**	**Conex**	**Pacot**	**Conex**	**Pacot**	
Medição 11	6	327.707	6	327.730	6	327.842	327.760
Medição 12	6	327.810	6	327.819	6	328.097	327.909
Medição 13	6	327.822	6	327.950	6	328.101	327.958

Os dados obtidos nos testes estão exibidos na Tabela 6.6, onde pode-se detectar que para cada medição feita, houve uma variação no número médio de pacotes capturados pelo monitin nos diferentes computadores, sendo ainda este valor, acima do número de pacotes injetados pelo tcpreplay. Acredita-se que esta diferença se deva a pacotes enviados por outros computadores, para o *host* alvo do monitoramento, durante a injeção de pacotes. Isso é possível, pois o ambiente de testes está conectado na intranet, e mesmo que o alvo esteja desconectado, algum outro computador poderá enviar um pedido de abertura de conexão a qualquer momento.

Apesar da variação no número de pacotes capturados, o resultado gerado pelo monitin não foi comprometido, pois ao final da injeção de pacotes, permaneceu a informação esperada, de seis conexões ativas.

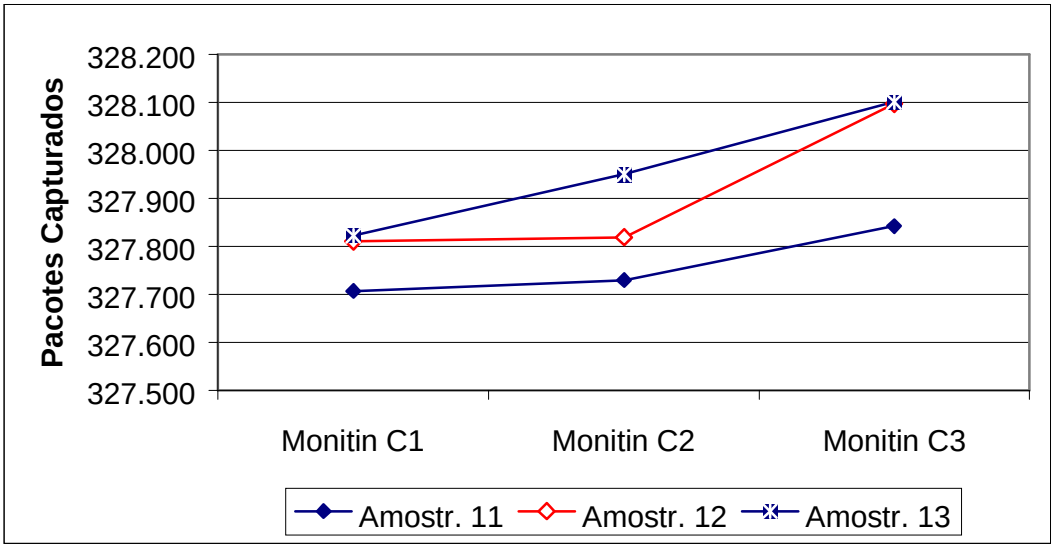

FIGURA 6.4 - Pacotes capturados durante o monitoramento de conexões

Outra constatação possível é que o computador Comp2, que possui mais memória RAM que Comp1, obteve melhor desempenho quanto ao número de pacotes capturados. Porém, a contagem final do número de conexões ativas não é comprometida. A Figura 8.4 ilustra esta constatação. Esta figura ainda permite visualizar que, para cada medição, onde todos os computadores estavam conectados no mesmo domínio de colisão, houve variação no número de pacotes capturados. Mesmo que neste caso isso não tenha refletido em distorções no resultado final, referente ao número de conexões ativas, é um indicativo de que isso poderá vir a ocorrer. Se o pacote perdido carregar informações de encerramento de conexões, então o software de monitoramento poderá continuar apresentando como ativa, uma conexão já encerrada.

Por isso, quanto menos pacotes perdidos, melhor será a resposta da ferramenta. Neste sentido, é importante observar na Figura 6.4, que a curva que representa a melhora na performance, obtida quando se aumenta a capacidade de processamento (Comp2 – Comp3) é muito mais acentuada do que a obtida quando existe apenas um aumento de capacidade de memória no equipamento (Comp1 - Comp2).

7 Conclusão

As atividades relacionadas com a administração de uma rede exigem o uso de uma série de ferramentas de apoio. O mercado oferece uma variedade delas fora do contexto de *software* livre. São pacotes fechados e em geral, com um custo elevado. No ramo do *software* livre, existe também uma gama de oferta nesse sentido. As principais vantagens observadas desta última categoria dizem respeito ao menor custo e a possibilidade de modificação do código fonte, a fim de embutir novas características na ferramenta.

Em contrapartida, efetuar alterações em uma ferramenta dessas, exige um bom conhecimento de programação e manipulação de dados em nível de bits e bytes, além, é claro, de um domínio sobre os protocolos e tecnologias de rede.

Todavia, essas características não podem ser abolidas. Para facilitar os mais leigos, o que pode ser feito é a utilização de diversas ferramentas, com o objetivo de cercar uma determinada situação. Isto é, utilizando as características mais interessantes de cada ferramenta, o administrador pode atingir seu objetivo de monitorar algum comportamento específico.

Em certas situações, quando as ferramentas oferecidas não refletem exatamente as necessidades do administrador da rede, não existe outra solução, senão, partir para a implementação. Neste estudo fez-se isso.

Foi construído o protótipo de uma ferramenta, o monitin, que permite exibir na console, as conexões ativas em cada instante de tempo, fazendo algo que as ferramentas estudadas não oferecem. As ferramentas vistas que mais se aproximam desta característica são sniffit, ethercap, ipaudit e especialmente iptraf. Os dois primeiros monitores exibem informações das conexões na console, porém o ethercap continua exibindo uma conexão, mesmo após o seu encerramento. Já o sniffit e o iptraf, que utilizam ncurses, apresentam deficiência quanto às opções de filtragem. O ipaudit, por outro lado, gera as informações de cada conexão em um arquivo de saída (em disco), onde é preciso analisar os tempos de abertura e fechamento de conexão para identificar se a conexão está ativa ou encerrada.

O monitin não apresenta a possibilidade de visualizar o conteúdo que está trafegando, somente a quantidade de bytes trocados, enquanto o sniffit e o ethercap, permitem. Isso, porém, não se trata de uma deficiência, pelo contrário, essa característica pode ser interpretada como uma vantagem, uma vez que o monitin não invade a privacidade do conteúdo da comunicação.

Outra característica do monitin é a possibilidade de contabilizar o tráfego trocado entre *hosts* da rede. Essa função permite que o administrador identifique quais são os computadores que estão gerando ou recebendo mais tráfego na rede. Existem diversas ferramentas capazes de gerar estatísticas sobre o tráfego da rede. Algumas delas geram informações relacionadas aos diferentes protocolos, aos tamanhos dos pacotes, ou ainda, como o monitin, quanto aos *hosts* que mais geram tráfego na rede. Porém, as mais completas nesse sentido, como darkstat e ntop, apresentam deficiência quanto a filtragem dos pacotes a serem analisados.

Entre as ferramentas estudadas com este enfoque, o ipfm possui características que combinadas com o monitin, produzem um bom resultado. O ipfm identifica somente os *hosts* que mais geram ou recebem dados, não apontando a origem ou destino dos mesmos. Isto é, o ipfm pode demonstrar, por exemplo, que o computador X foi o

que mais recebeu dados, mas não tem a capacidade de identificar que os enviou. Este é um exemplo clássico do uso das duas ferramentas conjugadas, para atingir um objetivo específico. Após identificar o computador "crítico" na rede com o ipfm; com o monitin é possível analisar a origem dos dados e as portas que possam estar vinculadas a tal comunicação. Esta característica se fez muito útil, para que se pudesse filtrar acessos supérfluos na rede, tais como, uso de programas para download de vídeos e mp3, nos testes feitos.

Uma terceira característica implementada na ferramenta é a possibilidade de se selecionar campos dos protocolos IP e TCP para serem exibidos na console. Selecionados os campos, para cada pacote recebido pela ferramenta (pacotes que passaram pelo filtro) são exibidos os dados correspondentes. Além de poder ser utilizada para alguma verificação ou depuração de um problema específico, esta função tem grande aproveitamento em nível acadêmico. Este recurso pode ser empregado para auxiliar a compreensão dos protocolos TCP/IP.

Por fim, é possível afirmar que o monitin é uma ferramenta útil no processo de administração da rede, seja no aspecto da segurança, ou em relação ao controle do tipo de tráfego existente.

TABELA 7.1 - Comparativo de Ferramentas de Captura de Pacotes

	Disponibiliza Acesso às Informações via HTTP	Realiza Estatísticas do Tráfego	Possui Recursos Avançados de Filtragem	Realiza Monitoramento de Conexões	Possui uma Abordagem Agressiva [3]	Permite Visualizar o Campo de Dados	Apresenta Saída Somente em Disco	Possui Saída em Modo Texto na Console	Utiliza NCurses	Utiliza Ambiente Gráfico	Lê Arquivos de *dump* de Tráfego	Gera Arquivos de *dump* de Tráfego
Darkstat	✔	✔								✔		
Ntop	✔	✔	✔							✔	✔	
Tcpdump			✔			✔		✔			✔	✔
Snort			✔					✔			✔	
Sniffit				✔		✔			✔		✔	✔
Ettercap		✔		✔	✔	✔			✔		✔	✔
Angst					✔		✔					✔
Ethereal		✔	✔			✔		✔		✔	✔	✔
Etherape		✔		✔						✔	✔	
Tcpflow			✔			✔	✔				✔	
Ipfm		✔					✔					
Ipaudit		✔		✔			✔				✔	✔
Iptraf		✔		✔					✔			
Monitin		✔	✔	✔				✔				

[3] Abordagem agressiva, neste contexto, significa que a ferramenta pode interferir no funcionamento normal da rede. Nestes casos, métodos ilícitos são utilizados para capturar pacotes em um ambiente conectado com *switch*.

Conforme pode ser constatado nos testes feitos, o monitin apresenta uma saída mais precisa, quando a trata de um foco restrito do ambiente. Isto é, em uma rede com muitos computadores, ao invés de monitorar todos as conexões existentes, ou todo o tráfego, é melhor utilizar o monitin fazendo uso de uma expressão de filtragem de pacotes. Desta forma, se estará restringindo a quantidade de pacotes a serem tratados pelo *software* aplicativo, diminuindo assim, a possibilidade de perda de pacotes.

Para realizar tal seleção de pacotes o monitim faz uso dos recursos de filtragem oferecidos pela libpcap. Tais recursos, que seguem o mesmo padrão utilizado pelo tcpdump, permitem a montagem de expressões de filtragem avançadas, combinando diversos atributos.

De forma diferente do ethereal e etherape, o monitin não exige ambiente gráfico para ser executado. Não permite exibição de dados visuais, mas em compensação a exigência de processador e memória é menor, e pode ser utilizado em um computador que utilize somente o linux em modo texto.

As ferramentas tcpflow, ethereal, ethercap, sniffit, tcpdump apresentam características que permitem a visualização dos dados embutidos nos pacotes IP. O tcpdump, de forma mais rústica. O tcpflow monitora o tráfego e monta o fluxo de dados em um arquivo em disco, para cada conexão detectada. Os demais fazem uso de recursos gráficos. O monitin foi construído sem a possibilidade de proporcionar a visualização dos dados, por entender-se que tal ação não se faz necessária e reflete uma invasão. Assim, o monitin pode ser liberado para ser utilizado por diferentes níveis hierárquicos da administração da rede, sem que se tenha nenhum risco de quebra de privacidade.

As ferramentas ntop e darkstat possuem um recurso muito interessante de acessibilidade das informações coletadas. Através de um servidor web oferecido, permitem que os dados sejam acessados a partir de qualquer ponto da rede, simplesmente fazendo uso de um navegador. Sob este aspecto, o monitin, só apresenta o recurso de visualização na console. A Tabela 7.1 ilustra as comparações entre as diversas ferramentas de captura de pacotes.

A experiência acumulada durante o desenvolvimento deste trabalho, indicou que é possível se construir ferramentas para captura de pacotes, de forma simples, quando apoiada sobre uma biblioteca de captura e filtragem, neste caso a libpcap. Porém, a dificuldade está em construir um código otimizado o suficiente, de forma a produzir um software que seja rápido o bastante para manipular as rajadas de pacotes que podem ocorrer na rede, com pequenas perdas.

Os testes demonstraram que o tcpdump, que já está consolidado como uma ferramenta de coleta bruta de dados da rede (*dump*), é mais eficiente que o monitin. Uma vez que foram construídos sobre a mesma biblioteca de captura, conclui-se que o tcpdump possui um código mais otimizado.

Desta forma, caso haja a intenção de coletar dados da rede (*dump*) para processamento posterior, sendo utilizado como entrada para algum outro processo, o tcpdump é a melhor escolha, se comparado como o monitin, na situação atual. Esta constatação é percebida especialmente quando o computador que está coletando os dados possuir processador de menor capacidade. Nos testes feitos, a diferença entre as duas ferramentas, atingiu um pico de 3,6% no número de pacotes capturados, com processador Pentium III de 1 Ghz. Os testes revelaram que quanto maior a capacidade do processador, mais as ferramentas se aproximam. Destes dados surge o indício de

problemas na otimização do código do monitin, pois para computadores com as mesmas características de processador, o computador com mais memória não apresenta melhorias expressivas no uso do monitin. Tal melhoria fica mais evidente quando existe um *upgrade* de processador.

Enfim, pode-se afirmar que este trabalho teve sucesso, à medida que com ele se obteve uma ferramenta que apresenta algumas características diferenciadas em relação a outras ferramentas estudadas. A principal vantagem, dentro da idéia inicial do estudo, é a possibilidade de monitorar transações realizadas sobre serviços internet, de forma *on-line*, em uma rede Ethernet. Esta característica, que é dada pelo monitoramento de conexões TCP, opera aliada a possibilidade da utilização de expressões de filtragem, o que não ocorre com as demais ferramentas semelhantes.

Em trabalhos futuros, poderia ser feita uma busca da melhor performance da ferramenta aqui prototipada. Com tal tarefa, a intenção é obter um melhor rendimento da mesma, permitindo que ela seja utilizada como meio de captura de pacotes para aplicações mais complexas, que exijam alta precisão. Dentro da idéia de servir como uma ferramenta de apoio acadêmico, que possa ser utilizada para o ensino de protocolos, a dissecação de outros protocolos poderia ser implementada. O monitin foi concebido com a finalidade de realizar um monitoramento *on-line*, portanto, não apresenta opções para receber como entrada, um *dump* de rede. Não apresenta também, uma funcionalidade que permita gerar tais arquivos de *dump*. Porém, tais características podem ser implementadas mediante o uso de funções específicas, disponíveis junto a libpcap.

Como nos testes feitos o próprio tcpdump apresentou variação na quantidade de pacotes capturados, devolvendo um valor não confiável referente ao número de pacotes perdidos, é interessante que se faça investigações sobre o *kernel*. Tais estudos, podem servir para buscar uma compreensão aprofundada sobre o processo de captura em nível de *kernel* e permitir buscar soluções ou melhorias para os problemas existentes atualmente, especialmente quando se deseja obter o valor exato do número de pacotes “perdidos”.

Anexo 1 Tráfego durante a validação do mecanismo de captura

No anexo 1 estão as figuras que ilustram o perfil do tráfego existente durante o período em que foi realizada as medições de 1 à 6, referentes à validação da capacidade de captura de pacotes, por parte do monitin.

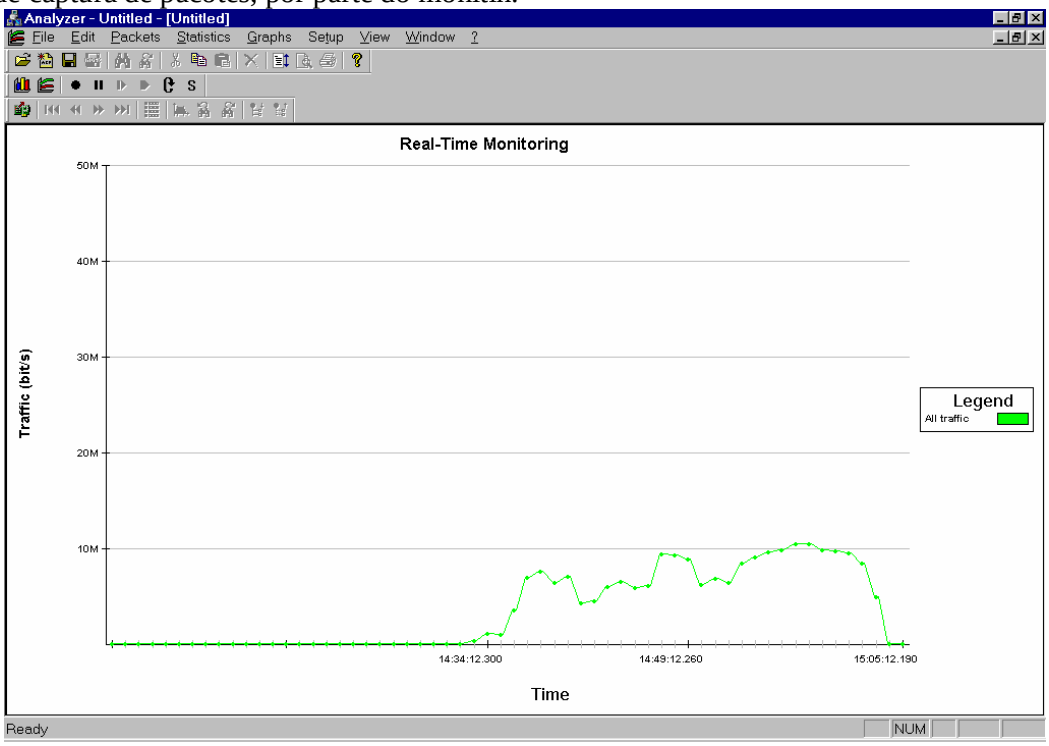

FIGURA 7.1 - Tráfego durante amostragem 1

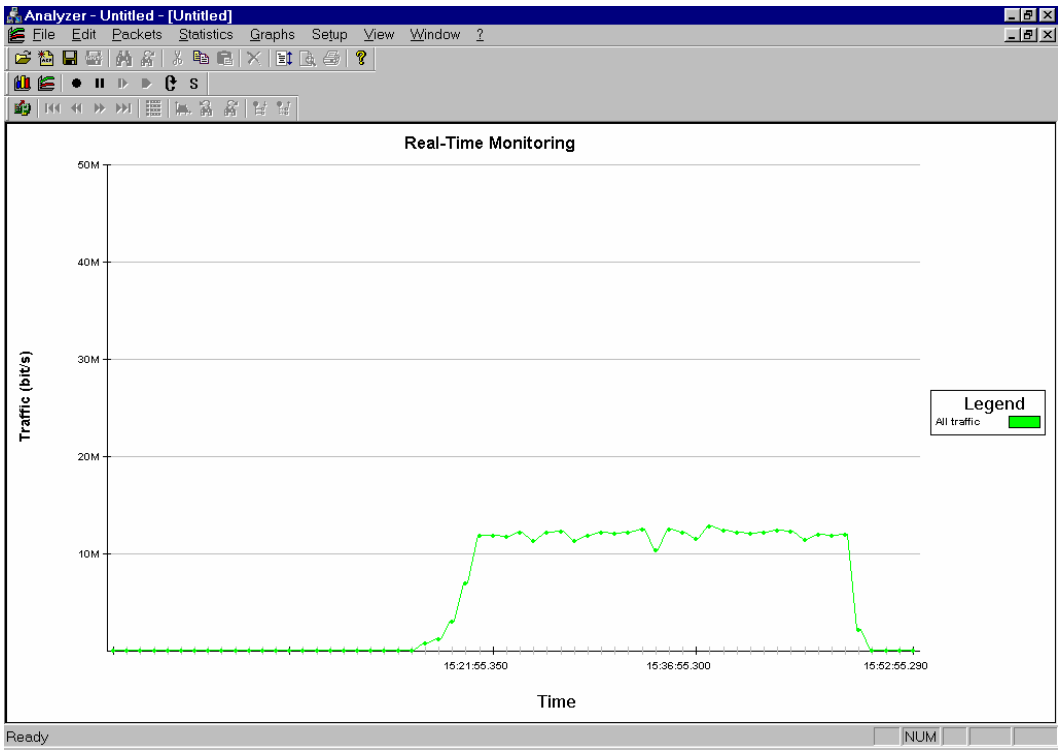

FIGURA 7.2 - Tráfego durante amostragem 2

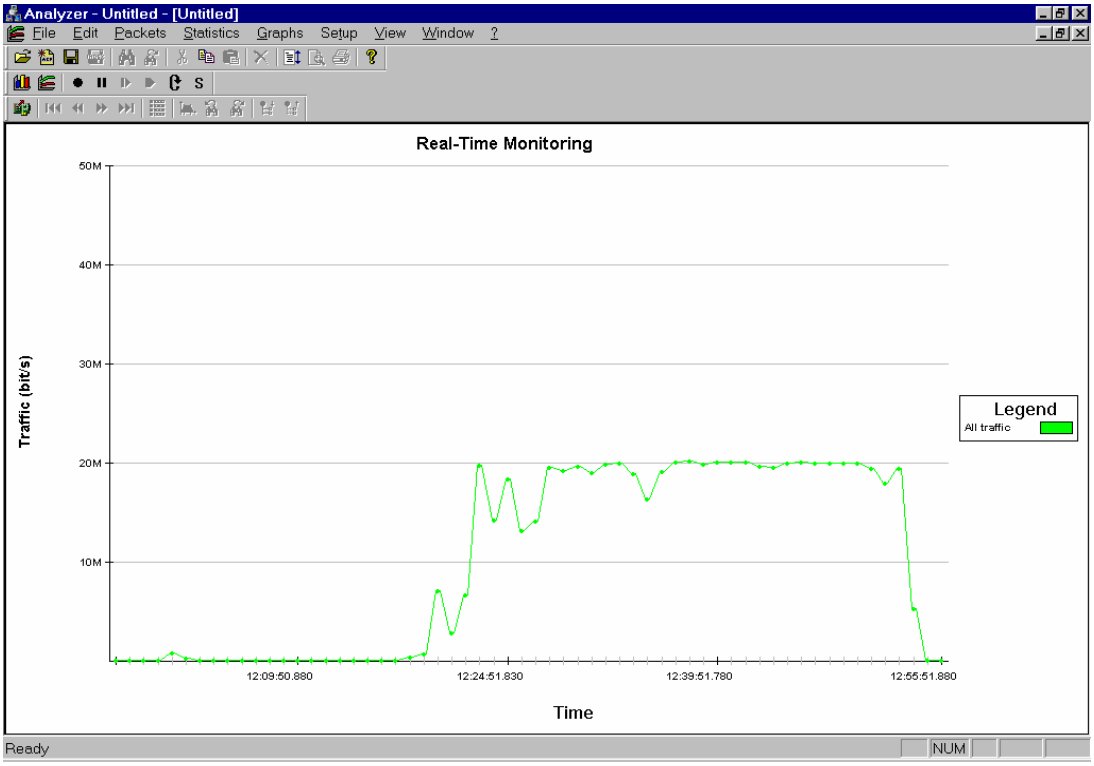

FIGURA 7.3 - Tráfego durante amostragem 3

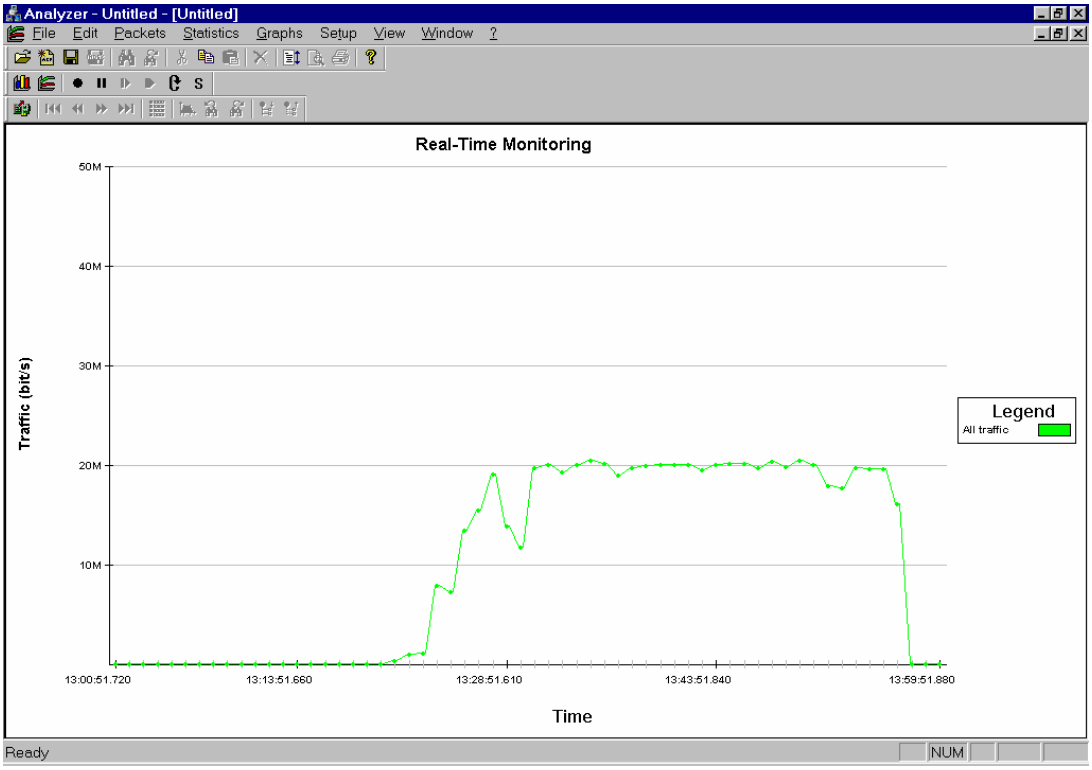

FIGURA 7.4 - Tráfego durante amostragem 4

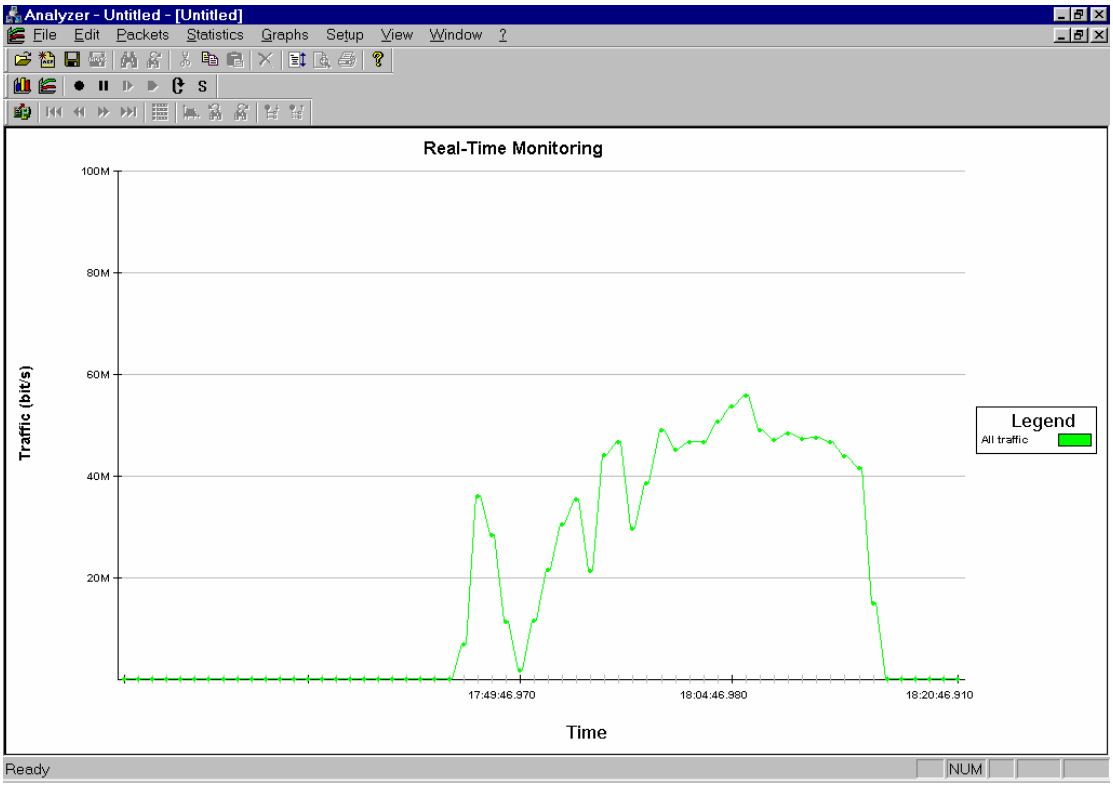

FIGURA 7.5 - Tráfego durante amostragem 5

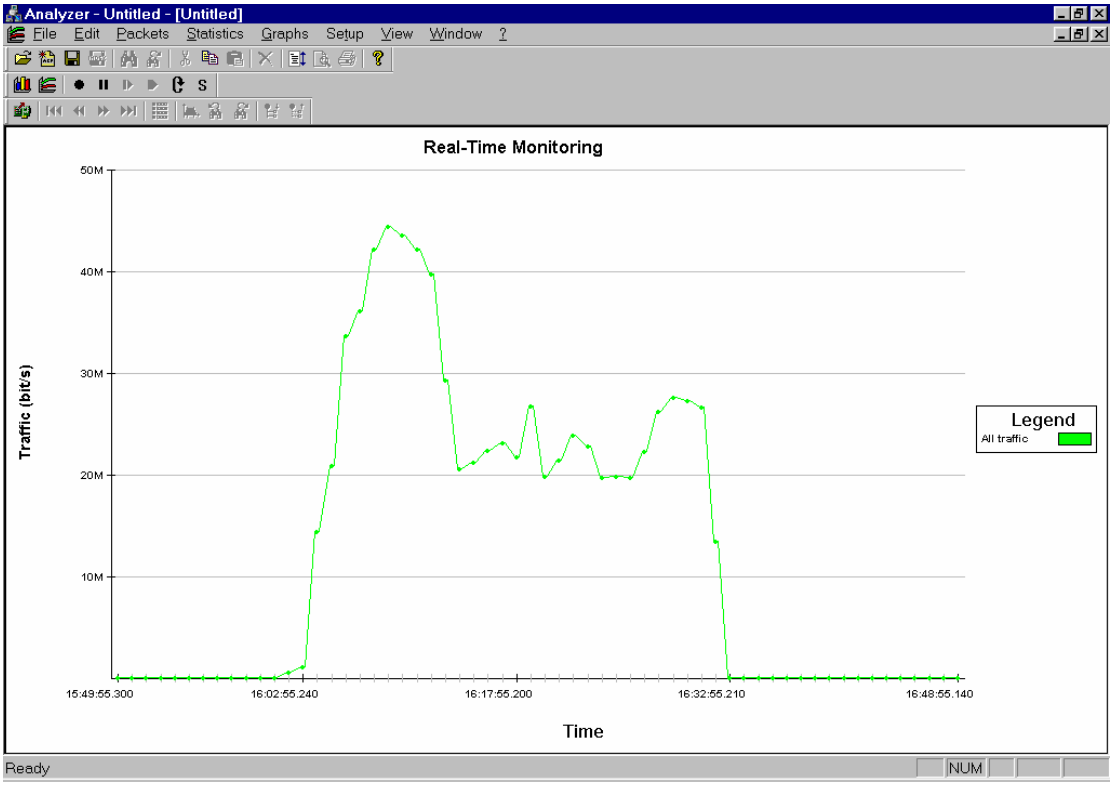

FIGURA 7.6 - Tráfego durante amostragem 6

Anexo 2 Tráfego durante a validação do algoritmo - Etapa 1

No anexo 2 estão as figuras que ilustram o perfil do tráfego existente durante o período em que foi realizada as medições 7 à 10, referentes à validação do algoritmo de monitoramento de conexões, em sua primeira etapa.

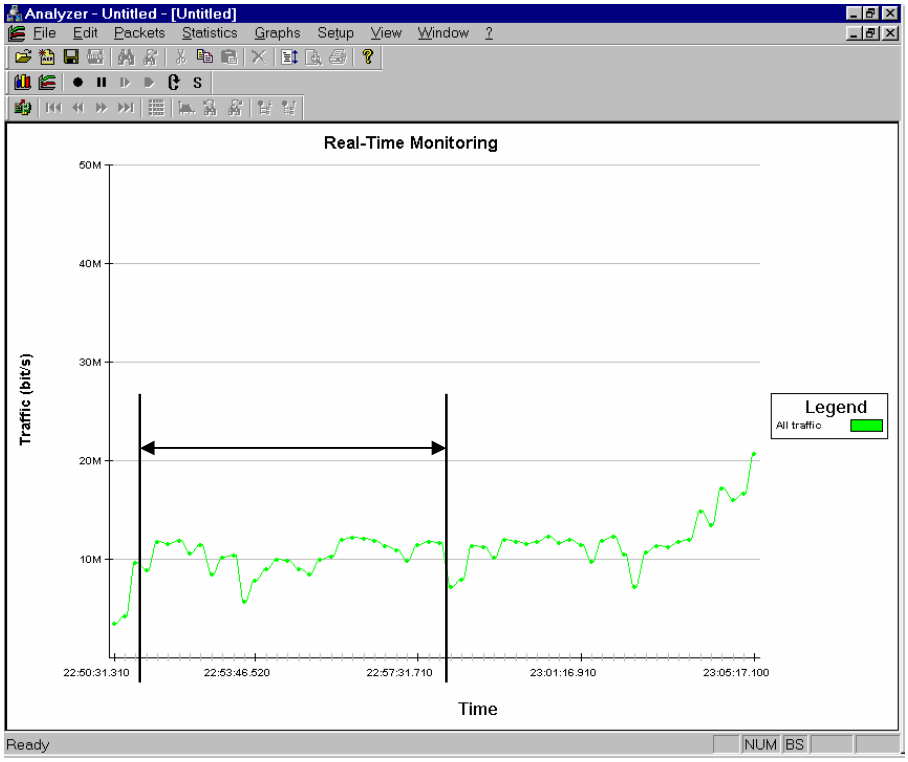

FIGURA 7.7 - Tráfego durante amostragem 7

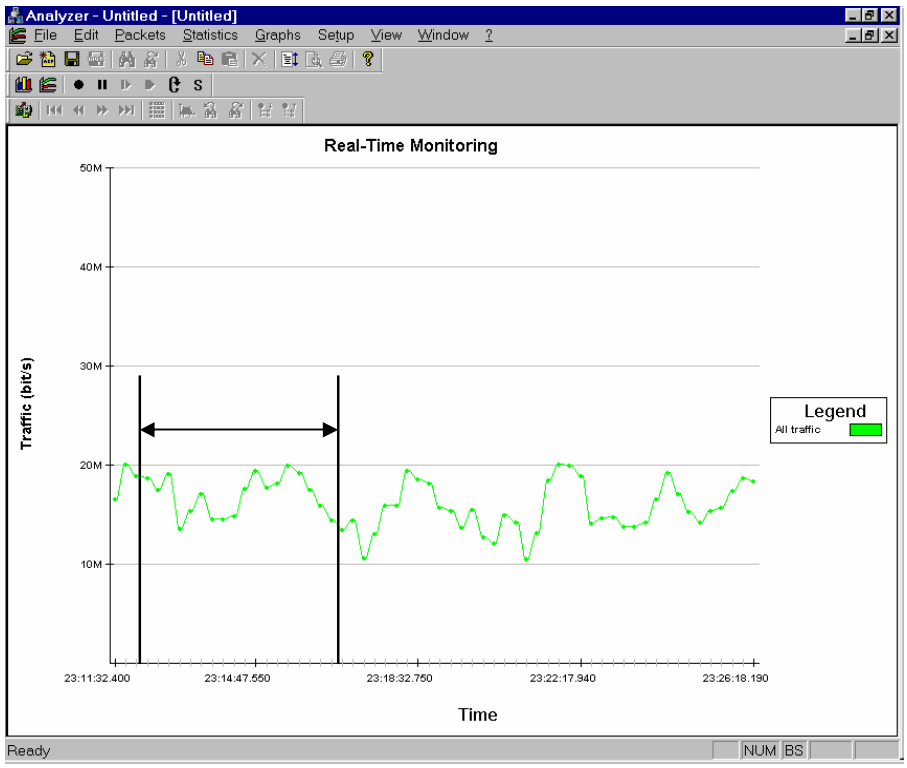

FIGURA 7.8 - Tráfego durante amostragem 8

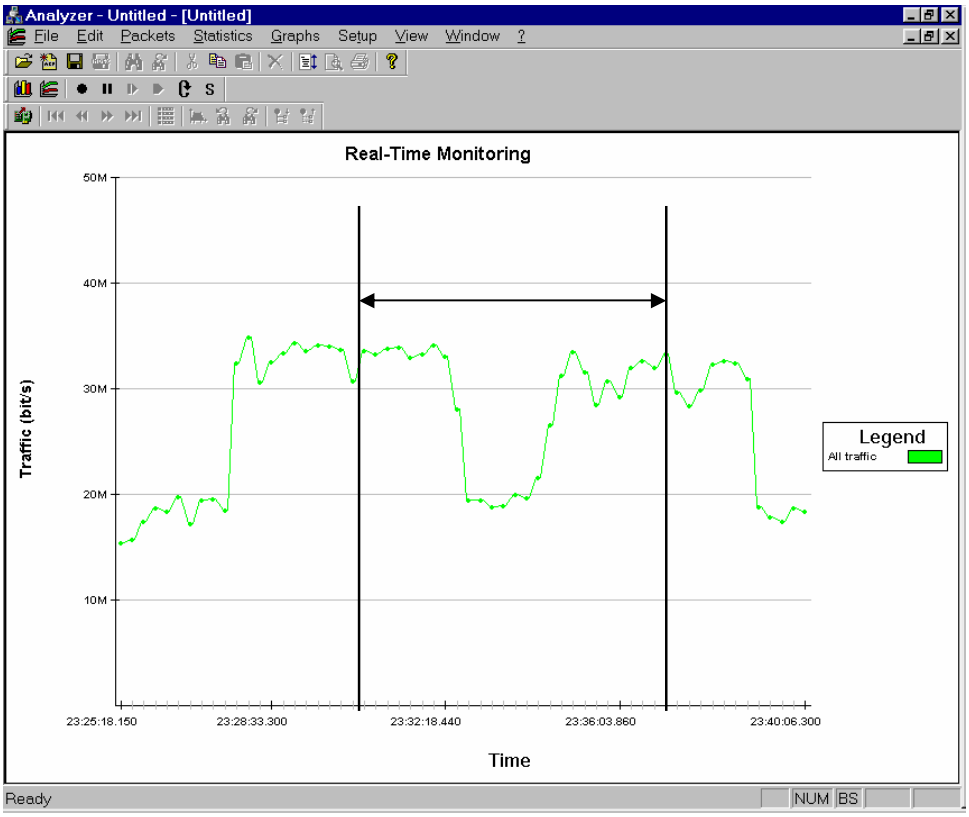

FIGURA 7.9 - Tráfego durante amostragem 9

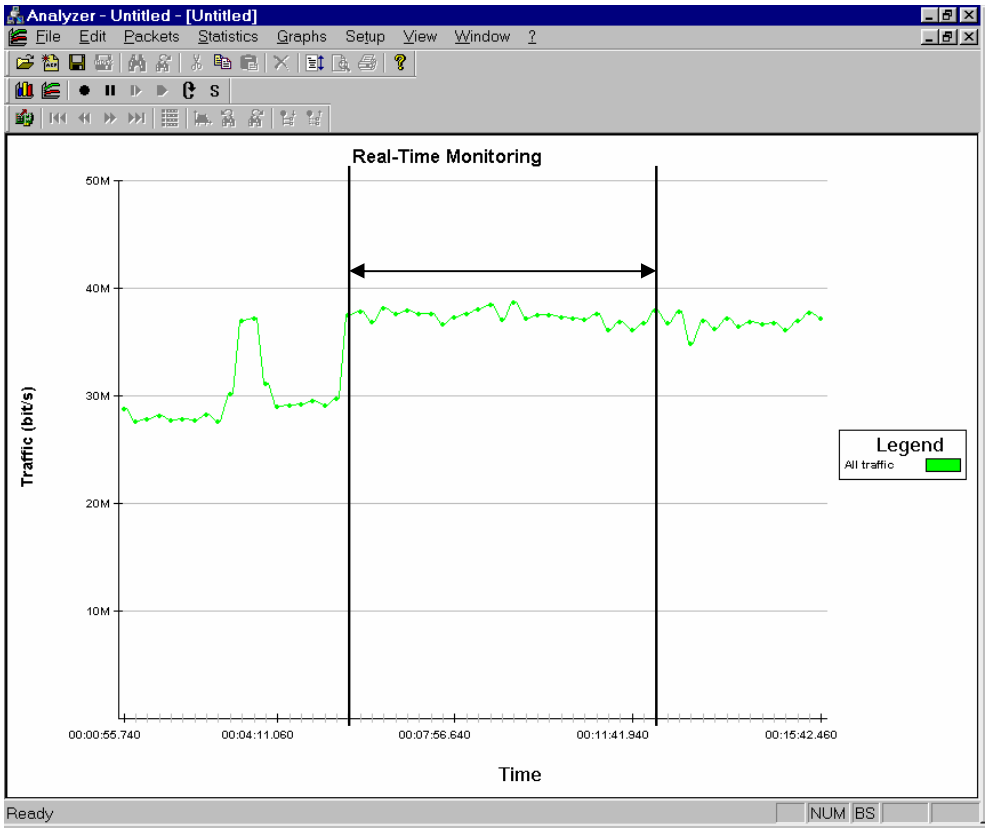

FIGURA 7.10 - Tráfego durante amostragem 10

Anexo 3 Tráfego durante a validação do algoritmo - Etapa 2

No anexo 3 estão as figuras que ilustram o perfil do tráfego existente durante o período em que foi realizada as medições 11 a 13, referentes à validação do algoritmo de monitoramento de conexões, em sua segunda etapa.

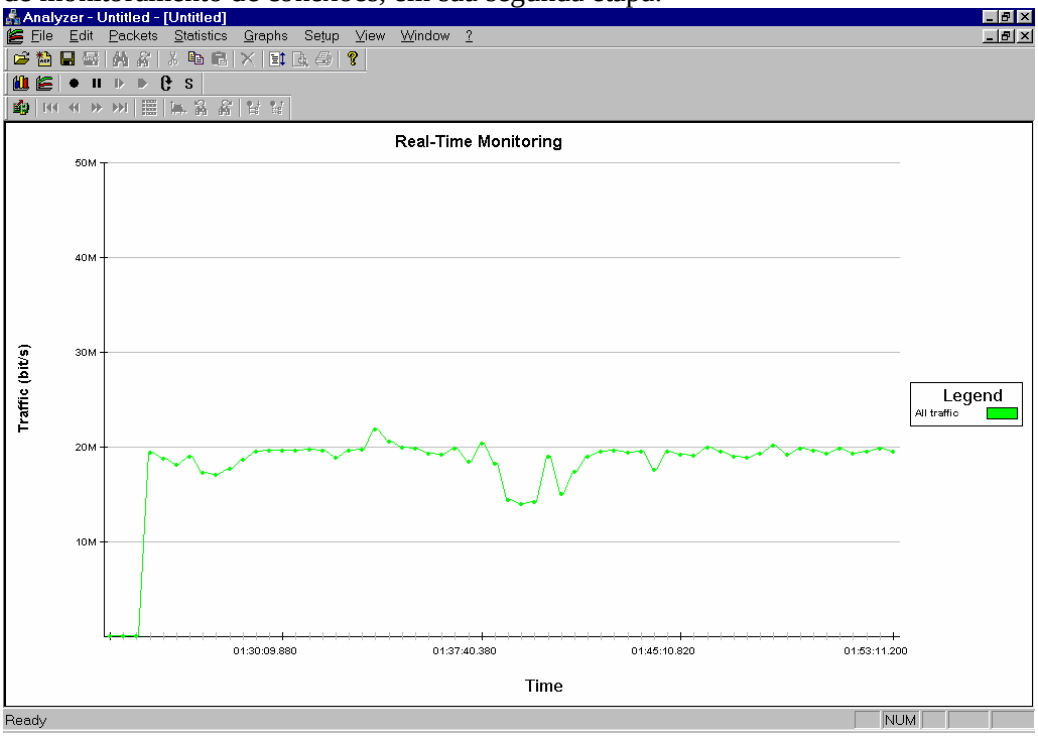

FIGURA 7.11 - Tráfego durante amostragem 11

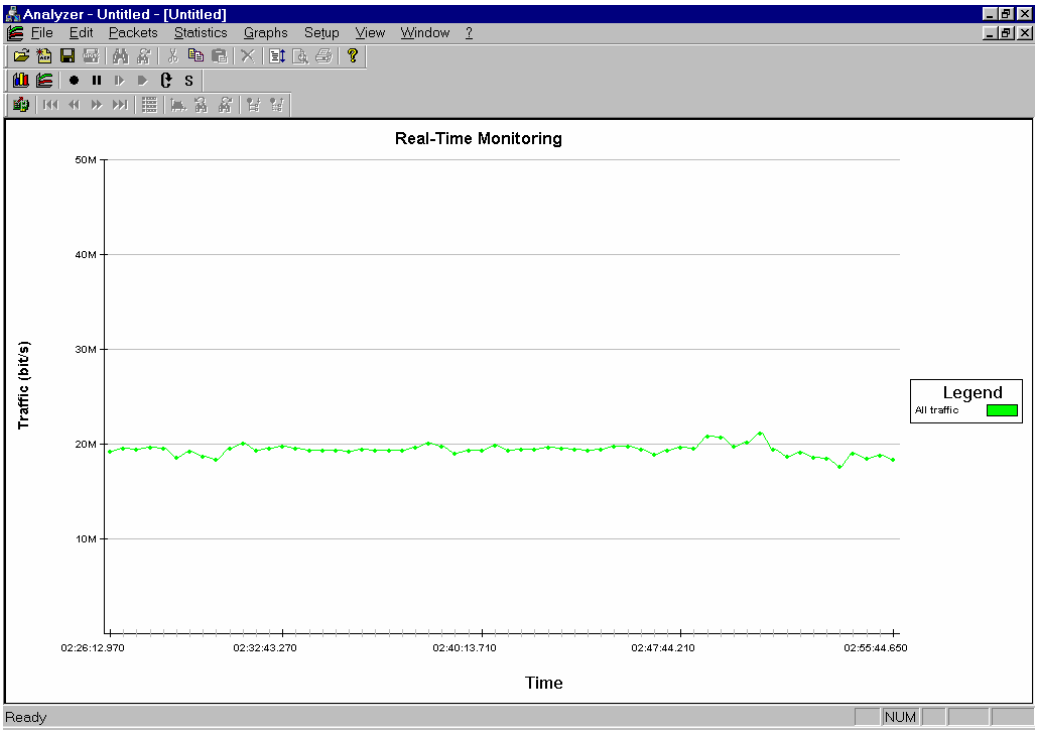

FIGURA 7.12 - Tráfego durante amostragem 12

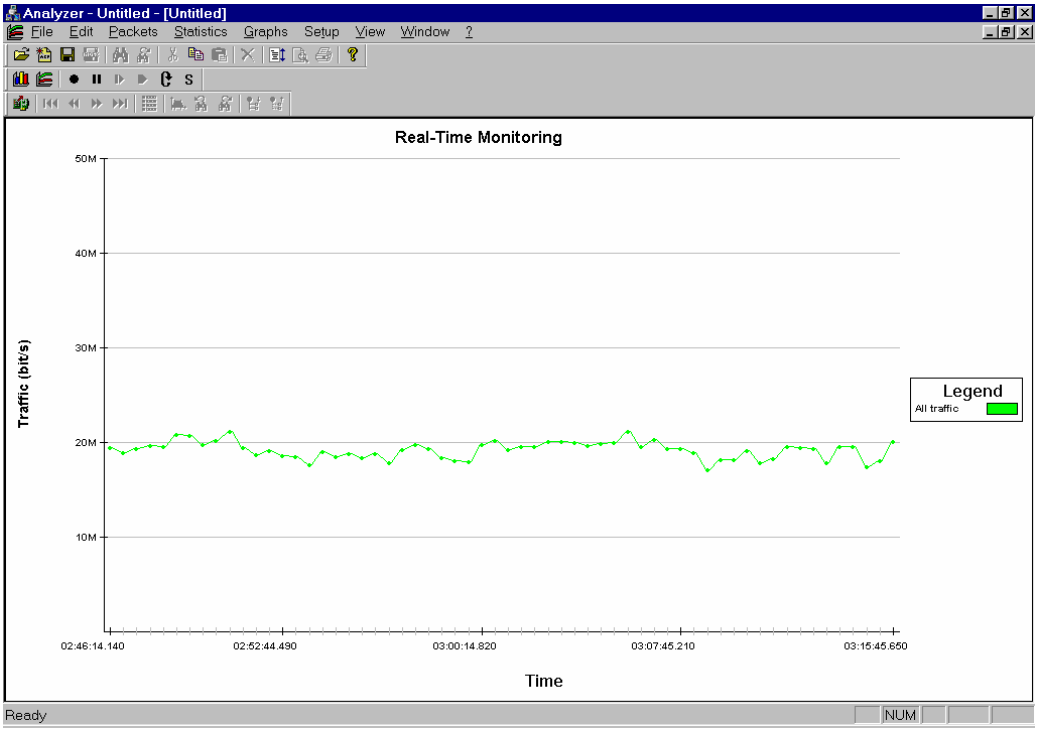

FIGURA 7.13 - Tráfego durante amostragem 13

Bibliografia

[ALB 2001] ALBUQUERQUE, Fernando. **TCP/IP – Internet:** protocolos & tecnologias. 3. ed. Rio de Janeiro:Axcel Books, 2001.

[ANG 2001] ANGST Manual. Disponível em: <http://angst.sourceforge.net>. Acesso em: 5 dez. 2001.

[BAR 98] BARR, Bob; YOO, Sung; CHEATHAM, Tom. Network Monitoring System Design. **SIGCSE Bulletin**, New York, v.30, n.1, p.102-106, Mar. 1998.

[BRA 89] BRADEN, R. **Requirements for Internet Hosts - Communication Layers**: RFC 1122. 1989. Disponível em: <http://www.ietf.org/rfc/rfc1122.txt?number=1122>. Acesso em: 25 jan. 2001.

[BRA 89a] BRADEN, R. **Requirements for Internet Hosts - Application and Support**: RFC 1123. 1989. Disponível em: <http://www.ietf.org/rfc/rfc1123.txt?number=1123>. Acesso em: 25 jan. 2001.

[BRE 99] BRENTON, Chris. **Mastering Network Security.** Alameda: Sybex Network Press, 1999.

[CAR 2001] CARTY, Aidan. **Building An IDS Solution Using Snort**. Disponível em: <http://packetstorm.decepticons.org/papers/IDS/snort4-latest.pdf>. Acesso em: 2 jul. 2001.

[CHA 95] CHAPMAN, Brent; ZWICKY, Elizabeth D. **Building Internet Firewalls**. Sebastopol: O'Reilly & Associates, 1995.

[COM 2001] COMER, Douglas E. **Redes de Computadores e Internet**. 2. ed. Porto Alegre: Bookman, 2001.

[COM 98] COMER, Douglas E. **Interligação de Redes com TCP/IP**. 2. ed. Rio de Janeiro: Campus, 1998.

[DAR 2001] DARKSTAT Manual. Disponível em: <http://members.optushome.com.au/emikulic/net/darkstat>. Acesso em: 21 fev. 2001.

[ETA 2001] ETHERAPE Manual. Disponível em: <http://etherape.sourceforge.net>. Acesso em: 2 dez. 2001.

[ETH 2001] ETHEREAL Manual. Disponível em: <http://www.ethereal.com/ethereal.1.html>. Acesso em: 11 abr. 2001.

[ETT 2001] ETTERCAP Manual. Disponível em: <http://ettercap.sourceforge.net>. Acesso em: 26 mar. 2001.

[FRE 2001] FREE Software Foundation. Disponível em: <http://www.fsf.org>. Acesso em: 20 jun. 2001.

[GAR 96] GARFINKEL, S.; SPAFFORD, G. **Practical Unix and Internet Security**. 2nd ed. Sebastopol: O'Reilly & Associates, 1996.

[GON 2000] GONÇALVES, Marcus. **Firewalls – A Complete Guide**. New York: McGraw-Hill, 2000.

[GRA 2000] GRAHAM, Robert. **Sniffing (network wiretap, sniffer) FAQ**. Disponível em: <http://www.robertgraham.com/pubs/sniffing-faq.html>. Acesso em: 15 jan. 2000.

[HAR 96] HARE, Chris; SIYAN, Karanjit. **Internet Firewalls and Network Security**. 2nd ed. [S.l.]: New Riders Publishing, 1996.

[HAT 2002] HATCH, Brian; LEE, James; KURTZ, George. **Hackers Linux Expostos**. São Paulo: Makron Books, 2002.

[HEL 99] HELD, Gilbert. **Comunicação de Dados**. Rio de Janeiro: Campus, 1999.

[HOR 84] HORNING, Charles. **A Standard for the Transmission of IP Datagrams over Ethernet Networks**: RFC 894. 1984. Disponível em: <http://www.ietf.org/rfc/rfc0894.txt?number=894>. Acesso em: 10 mar. 2001.

[HUN 94] HUNT, Craig. **TCP/IP Network Administration**. Sebastopol: O'Reilly & Associates, 1994.

[INS 2001] INSOLVIBILE, Gianluca. Linux Socket Filter: Sniffing Bytes Over The Network. **Linux Journal,** Seatle, n. 86, June 2001. Disponível em: <http://www.linuxjournal.com/article.php?sid=4659>. Acesso em: 25 mar. 2002.

[IPA 2001] IPAUDIT Manual. Disponível em: <http://ipaudit.sourceforge.net>. Acesso em: 22 nov. 2001.

[IPF 2001] IPFM Manual. Disponível em: <http://robert.cheramy.net/ipfm>. Acesso em: 18 mar. 2001.

[IPT 2001] IPTRAF Manual. Disponível em: <http://cebu.mozcom.com/riker/iptraf>. Acesso em: 22 nov. 2001.

[McC 93] McCANNE, Steven; JACOBSON, Van. The BSD Packet Filter: a New Architecture for User-Level Packet Capture. In: USENIX CONFERENCE, 1993. **Proceedings...** [S.l.: s.n.], 1993. p.259-269.

[MER 2000] MERWE, Jacobus Vander; CÁRCERES, Ramón; CHU, Yang-hua; SREENAN, Cormac. mmdump: a Tool for Monitoring Internet Multimedia Traffic. **ACM SIGCOMM Computer Communication Review,** New York, v.30, n.5, p.48-59, Oct. 2000.

[MOG 85] MOGUL, J.C.; POSTEL, Jon. **Internet Standard Subnetting Procedure**: RFC 950. 1985. Disponível em: <http://www.ietf.org/rfc/rfc0950.txt?number=950>. Acesso em: 16 fev. 2001.

[NAK 2002] NAKAMURA, Emílio Tissato; GERUS, Paulo Lício de. **Segurança de Redes em Ambientes Corporativos**. São Paulo: Berkeley Brasil, 2002.

[NOV 2001] NOVELL Support Documentation. Disponível em: <http://support.novell.com>. Acesso em: 20 jun. 2001.

[NTO 2001] NTOP Documentation. Disponível em: <http://www.ntop.org>. Acesso em: 5 mar. 2001.

[ORA 2001] ORACLE-Linux Center. Disponível em: <http://technet.oracle.com/tech/linux/content.html>. Acesso em: 20 jun. 2001.

[PLU 82] PLUMMER, David C. **An Ethernet Address Resolution Protocol**: RFC 826. 1982. Disponível em: <http://www.ietf.org/rfc/rfc0826.txt?number=826>. Acesso em: 15 fev. 2001.

[POS 80] POSTEL, J. **User Datagram Protocol**: RFC768. 1980. Disponível em: <http://www.ietf.org/rfc/rfc0768.txt?number=768>. Acesso em: 25 jan. 2001.

[POS 81] POSTEL, Jon. **Internet Protocol**: RFC 791. 1981. Disponível em: <http://www.ietf.org/rfc/rfc0791.txt?number=791>. Acesso em: 25 jan. 2001.

[POS 81a] POSTEL, Jon. **Internet Control Message Protocol**: RFC 792. 1981. Disponível em: <http://www.ietf.org/rfc/rfc0792.txt?number=792>. Acesso em: 26 jan. 2001.

[POS 81b] POSTEL, Jon. **Transmission Control Protocol**: RFC 793. 1981. Disponível em: <http://www.ietf.org/rfc/rfc0793.txt?number=793> . Acesso em: 25 jan. 2001.

[REY 94] REYNOLDS, J; POSTEL, J. **Assigned Numbers**: RFC 1700. 1994. Disponível em: <http://www.ietf.org/rfc/rfc1700.txt?number=1700>. Acesso em: 26 jan. 2001.

[SIY 95] SIYAN, Karankit; HARE, Chris. **Internet Firewalls and Network Security**. Indianápolis: New Riders Publishing, 1995.

[SMI 2001] SMITH, F. Donelson; CAMPOS, Felix Hernander; JEFFAY, Kevin; OTT, David. What TCP/IP Protocol Hearders Can Tell Us About the Web. **ACM SIGMETRICS Performance Evaluation Review**, Cambridge, v.29, p.245-256, June 2001.

[SNI 2001] SNIFFIT Manual. Disponível em: <http://reptile.rug.ac.be/~coder/sniffit/sniffit.html>. Acesso em: 10 abr. 2001.

[SNO 2001] SNORT User Manual. Disponível em: <http://www.snort.org/docs>. Acesso em: 10 jan. 2001.

[SOA 95] SOARES, Luiz F. G.; LEMOS, Guido; COLCHER, Sérgio. **Redes de Computadores:** das LANS, MANS e WANS às Redes ATM. Rio de Janeiro: Campus, 1995.

[SOC 91] SOCOLOFSKY, T.J.; KALE, C.J. **TCP/IP tutorial**: RFC 1180. 1991. Disponível em: <http://www.ietf.org/rfc/rfc1180.txt?number=1180>. Acesso em: 20 jan. 2001.

[SPU 2000] SPURGEON, Charles E. **Ethernet:** o guia definitivo. Rio de Janeiro: Campus, 2000.

[TAN 97] TANENBAUM, Andrew S. **Redes de Computadores**. 4. ed. Rio de Janeiro: Campus, 1997.

[TAY 2001] TAYLOR, Carol; FOSS, Jim Alves. NATE – Network Analisis of Anomalous Traffic Events, a low-cost approach. In: WORKSHOP ON NEW SECURITY PARADIGMS, 2001. Cloudcroft, New Mexico. **Proceedings...** New York: ACM Press, 2001. p.89-96.

[TCP 2001] TCPDUMP Manual Page. Disponível em: <http://www.tcpdump.org/tcpdump_man.html>. Acesso em: 15 jan. 2001.

[TCF 2001] TCPFLOW Manual. Disponível em: <http://www.circlemud.org/pub/jelson/tcpflow/ChangeLog>. Acesso em: 14 abr. 2001.

[TET 2001] TETHEREAL Manual. Disponível em: <http://www.ethereal.com/tethereal.1.html>. Acesso em: 11 abr. 2001.

[TOR 2001] TORRES, Gabriel. **Redes de Computadores**: curso completo. Rio de Janeiro: Axcel Books, 2001.

[WEB 2000] WEBER, Raul Fernando. Segurança na Internet. In: JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA, 19., 2000, Curitiba. **Anais...** Curitiba: Champagnat, 2000. p.43-82.

[ZIE 2000] ZIEGLER, Robert L. **Linux Firewalls**. Indianápolis: New Riders Publishing, 2000.