

**UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO**

Madalena Pereira da Silva

ANALISADOR DE PROTOCOLO
Analisar e Capturar Pacotes TCP/IP

Prof. Dr. Alexandre Moraes Ramos

Lages (SC), maio de 2002

Sumário

1. Introdução
2. Analisador de Protocolos
 - 2.1 Funcionamento do Analisador
3. Seleção de Ferramentas
 - 3.1 Ethereal – Network Protocol Analyzer
 - 3.1.1 Características do Ethereal – Network Protocol Analyzer
 - 3.1.2 Bibliotecas: LibPCap e WinPCap
 - 3.2 Instalando o Analisador de Protocolo Ethereal
 - 3.3 Utilizando o Analisador de Protocolo Ethereal
 - 3.3.1 Barra de Menu do Ethereal
 - 3.3.2 Capturando informações com o Ethereal
 - 3.3.3 Visualizando as Informações Capturadas
 - 3.3.4 Aplicando Filtros a Exibição de Pacotes
4. Descrição do Ambiente
5. Análise dos Protocolos
 - 5.1 Protocolo IP
 - 5.1.1 Datagrama IP
 - 5.1.1.1 Campos do Datagrama IP
 - 5.1.2 Fragmentação IP
 - 5.1.3 Resolução de Endereços
 - 5.2 Protocolo TCP
 - 5.2.1 Formato do Segmento TCP
 - 5.2.1.1 Campo de um Segmento TCP
 - 5.2.2 Estabelecimento de Conexões TCP
 - 5.2.3 Múltiplas Conexões em uma mesma Porta
 - 5.2.4 Transferência de Dados no TCP
 - 5.2.4.1 Ausência de Criptografia
 - 5.2.4.2 Buffers, Controle de Fluxo e Janela
 - 5.2.5 Encerramento de Conexões TCP
- 6 Conclusão
- 7 Referências Bibliográficas

1 Introdução

Um analisador de protocolo é um dispositivo que pode ser configurado para contar ou apresentar os pacotes à medida que passam através de uma rede compartilhada.

Em virtude de os analisadores de protocolos serem capazes de capturar todos os protocolos da rede à medida que eles percorrem o cabeamento, qualquer pessoa pode acompanhar o seu funcionamento e ter acesso aos “valiosos” dados da rede. A maioria dos sistemas operacionais, criptografa informações como o nome e senha de usuário, mas não criptografa os dados.

Uma vez instalado um analisador de protocolo na rede, qualquer usuário pode visualizar os pacotes que trafegam, porém para realizar uma análise será necessário que o pesquisador tenha o conhecimento e o entendimento do funcionamento dos protocolos, de modo a possibilitar a compreensão das informações que estão sendo capturadas.

A análise do tráfego dos pacotes que circulam em uma rede demanda tempo e exige-se um conhecimento razoável do funcionamento dos protocolos que estão sendo analisados. A fase de análise vai além da visualização do tráfego, pois permite que o administrador tenha um feedback do que está ocorrendo na rede e se necessário for isolar eventuais falhas que impossibilitam o seu funcionamento adequado.

O trabalho tem por objetivo, através de um analisador de protocolos realizar a captura e análise do protocolo TCP/IP na rede local do Hemocentro Regional de Lages – HRL. Onde através das informações coletadas, pretende-se tecer algumas considerações que auxiliem no entendimento dos protocolos e demonstrem algumas características do tráfego da rede analisada.

2 Analisador de Protocolo

Um analisador de rede é um dispositivo que pode ser usado para depurar problemas em uma rede. Muitos analisadores são portáteis permitindo que sejam movidos facilmente. Após ser acoplado em uma rede, um analisador pode monitorar eventos específicos e pode relatar estatísticas como o número médio de quadros por segundo ou o tamanho médio de quadro. Mais importante: - maioria dos analisadores são flexíveis – um gerente pode configurar um analisador para observar os quadros enviados por uma máquina específica, observar tráfego de um tipo específico ou computar a porcentagem de quadros de cada tipo.

2.1 Funcionamento do Analisador

Um analisador consiste em um pequeno computador com hardware de interface de rede e software de analisador convencional. O programa analisador permite que um usuário configure parâmetros e então os utilize para analisar pacotes.

Para ler os pacotes, o software analisador coloca o hardware de interface de rede do computador em modo *promíscuo*, que anula o reconhecimento de endereço convencional. Isto é, o software do computador configura a interface de rede do computador para aceitar todos os quadros. Uma vez em modo promíscuo, a interface não verifica o endereço de destino, nem rejeita quaisquer quadros. Em vez disso a interface simplesmente coloca uma cópia de cada quadro na memória do computador e interrompe a CPU para informá-la que chegou um quadro.

Os usuários que não são informados podem assumir que uma placa de interface especial é exigida para modo promíscuo. Surpreendentemente, este não é o caso. Quase todo o hardware de interface de rede comercialmente disponível suporta leitura promíscua. Além disso, colocar uma interface de rede em modo promíscuo é normalmente trivial – apenas uma grande quantidade de instruções de CPU precisa ser executada. Como consequência qualquer usuário com um computador acoplado a uma LAN, pode ler todos os pacotes que viajam através da rede. Isto é, qualquer computador acoplado a uma LAN pode espiar toda comunicação; as mensagens enviadas através de uma LAN não têm garantia de serem privadas.

Um analisador de rede é configurável; a configuração exata que um usuário seleciona determina quais campos o analisador examina e que informações ele mantém.

Alguns softwares analisadores trabalham com um conceito chamado de “dissecadores de protocolo”. Os dissecadores são funcionalidades adicionadas ao software que possibilitam extrair dos dados capturados as informações inerentes ao protocolo contido nos mesmos. Para ficar mais claro, podemos citar o exemplo de um dissecador para o protocolo IP. Neste caso, um dissecador é capaz de extrair dos dados capturados, os pacotes IP existentes e interpretar os campos do pacote como, por exemplo, endereço IP de origem e destino.

3 Seleção da Ferramenta

Após a compreensão do funcionamento e da utilidade dos analisadores de protocolos, partiu-se para a escolha das ferramentas. Atualmente existe uma série de analisadores de protocolos disponíveis na *Web*, porém antes de selecionar a ferramenta ideal, uma avaliação foi efetuada em pelo menos cinco softwares analisadores de rede. Nesta análise foram considerados alguns requisitos tais como: portabilidade, flexibilidade, interface amigável com o usuário, facilidade de configuração, consistência e simplicidade na apresentação dos resultados.

Todos os analisadores de protocolos permitem filtrar e classificar os protocolos da rede que está sendo monitorada, no entanto alguns apresentam maior funcionalidades que outros. Com na avaliação dos softwares analisadores o *Ethereal – Network Protocol Analyzer* – versão 0.9.3, foi o analisador que se mostrou o mais adequado para a captura e análise do protocolo TCP/IP.

3.1 *Ethereal – Network Protocol Analyzer*

O *Ethereal* é um software analisador de protocolos gratuito, foi desenvolvido sob o sistema GNU (*General Public License*), pode ser alterado livremente, pois possui o código fonte aberto. A primeira versão do *Ethereal* foi desenvolvida por *Gerald Combs* em 1997. Hoje um grande número de pessoas estão trabalhando no seu desenvolvimento, corrigindo *bugs* e implementando novas funcionalidades.

O *Ethereal* é um analisador de protocolos para ambientes Unix e Windows. Permite ao administrador examinar os dados de uma rede em tempo real ou de um segmento específico capturado e salvo em disco. Possui características poderosas, como uma rica linguagem de filtragem e a habilidade de exibir a sequência reconstruída de uma sessão TCP.

3.1.1 Características do *Ethereal – Network Protocol Analyzer*

Segundo [*Ethereal*], abaixo segue uma lista das principais características desse software analisador:

- Permite capturar e analisar os dados de uma rede;

- Trabalha com arquivos gerados por outros softwares de captura como o Tcpdump, NetXRay, IPTrace, entre outros;
- Instalação tanto em estações UNIX quanto em Windows;
- Capaz de capturar dados de várias interfaces;
- Captura dados em modo promíscuo;
- Interpretação de uma gama muito grande de protocolos;
- Possibilita tanto a análise detalhada dos dados capturados quanto à criação de resumos dos dados coletados;
- Disponibiliza a utilização de grande número de filtros de captura;
- Possui um rico sistema de filtros de exibição que podem ser aplicados aos dados coletados, possibilitando assim a extração de pacotes específicos sem a necessidade de executar uma nova captura. É possível também salvar em um novo arquivo apenas os dados que são relevantes.
- Trabalha com o conceito de dissecadores de protocolo

3.1.2 Bibliotecas: LibPCap e WinPCap

O Software analisador de rede foi projetado para suportar as plataformas Unix e Windows. Quando instalado em uma estação UNIX o analisador faz uso de uma biblioteca chamada LibPCap para capturar os dados que trafegam pela rede. A biblioteca LibPCap possibilita a captura de informações da rede no nível do usuário e é utilizada por uma série de softwares disponíveis no ambiente UNIX.

A LibPCap proporciona uma estrutura para monitoramento de rede em baixo nível. Várias aplicações podem fazer uso desta biblioteca para prover uma série de informações, como estatísticas de rede, efetuar monitoramento de segurança, procurar erros na rede, etc. Desta forma, conforme descrito em [TCPPDump] é possível desenvolver softwares com interfaces diferentes no que se refere à apresentação dos dados e, utilizar a mesma biblioteca para captura das informações, através das APIs disponíveis na LibPCap. A LibPCap suporta um mecanismo de filtros de captura baseado na arquitetura de filtros BSD, descrito em 1993 no artigo Winter Usenix.

Apesar de proporcionar às aplicações uma certa independência de plataforma, a LibPCap continuava restrita às plataformas UNIX. Com o desenvolvimento de uma versão da LibPCap para o ambiente Windows (a WinPCap), várias aplicações antes só disponíveis no ambiente UNIX puderam ser portadas.

A biblioteca WinPCap foi desenvolvida para ser utilizada no sistema operacional Windows. Esta biblioteca está baseado na LibPCap e possibilitou portar para o ambiente Windows uma série de aplicações que antes só estavam disponíveis no ambiente UNIX.

As mesmas funcionalidades disponíveis no ambiente Unix, também estão disponíveis para o ambiente Windows. A WinPCap é composta de um driver em nível de kernel baseado no mesmo sistema de filtros utilizado no UNIX e por uma biblioteca em alto nível que proporciona várias funcionalidades necessárias aos desenvolvedores.

O driver de captura de pacotes adiciona aos sistemas operacionais da Microsoft, as mesmas funcionalidades de capturar as informações que trafegam pela rede de uma forma muito parecida com o filtro de pacotes BSD, disponível apenas no kernel de várias distribuições UNIX.

A biblioteca de ligação dinâmica *Packet.dll* proporciona uma API que pode prover o acesso as funções do driver de captura. A WinPcap também exporta uma série de funções compatíveis com a LibPCap, oferecendo assim um conjunto de funções de alto nível para a captura de pacotes, ocultando o hardware de rede ou o sistema operacional envolvidos.

3.2 Instalando o Analisador de Protocolo Ethereal

Para a instalação do Ethereal, antes se faz necessário instalar uma das bibliotecas: LibPCap ou WinPCap. Neste caso foi instalado a WinPCap, que é a biblioteca que o Ethereal utiliza para efetuar a captura de informações da rede no ambiente Windows.

Para a instalação da biblioteca WinPcap, inicialmente foi realizado o download da versão 2.3 – *WinPCap.exe* disponível no site: <http://netgroup-serv.polito.it/winpcap/install/Default.htm>. O arquivo possui 320KB. Assim que efetuar o download do arquivo, basta executá-lo e seguir os passos necessários para a instalação. Durante a instalação da WinPCap será exibido uma tela de boas vinda e uma tela de exibição do contrato, o qual deve ser aceito para que a instalação continue e seja concluída com uma tela de finalização.

Assim que for instalada a biblioteca, faz se necessário realizar o download do arquivo de instalação do Ethereal – versão 0.9.3 - *ethereal-setup-0.9.3-1.exe*. O arquivo possui aproximadamente 6.3MB e encontra-se disponível no endereço:

<http://www.ethereal.com/distribution/win32/ethereal-setup-0.9.3-1.exe>. No instante em que for realizado o download do arquivo, basta executá-lo e seguir os passos para efetuar a instalação.

3.3 Utilizando o Analisador de Protocolo Ethereal

A ferramenta Ethereal permite aos usuários fazer uso de seus utilitários para o modo comando ou modo gráfico. Nesta documentação será demonstrado às facilidades proporcionadas pela interface gráfica. No momento em que o usuário abrir o programa, será apresentado uma tela - janela principal do software, conforme ilustração da figura abaixo:

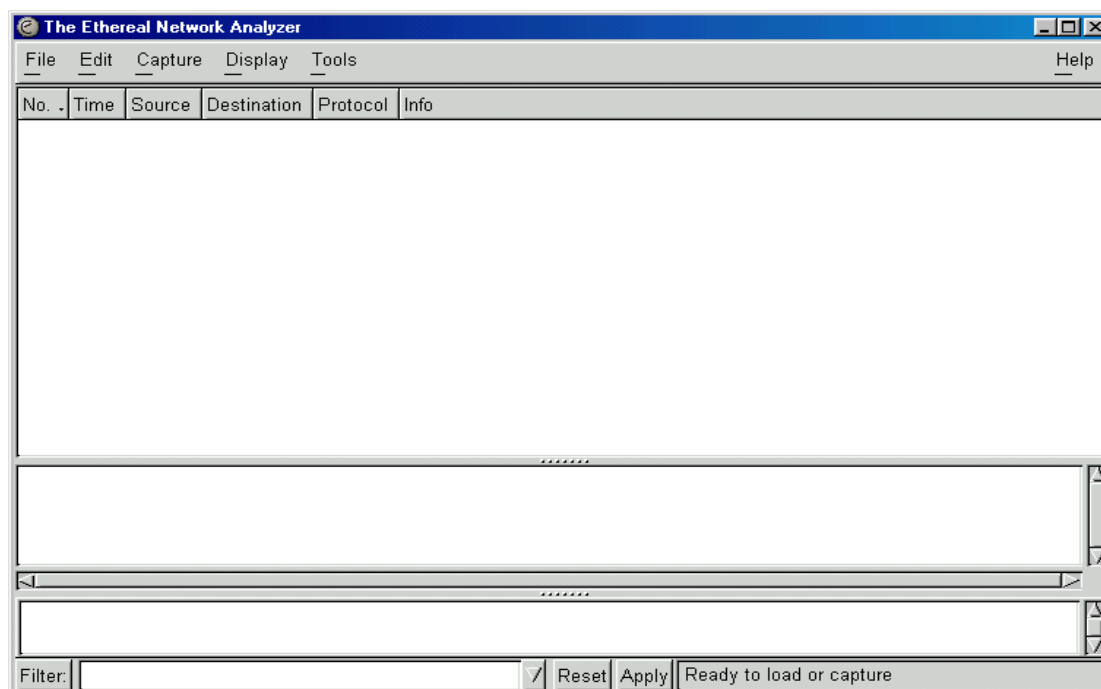


Figura 01 – Tela principal do Ethereal




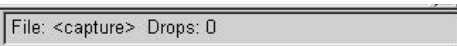
A janela principal do software Ethereal é composta por três importantes partes: superior, intermediária e inferior.

A parte superior da janela exibe um resumo em forma de lista dos pacotes que foram capturados. Para observar detalhes do pacote, selecionar a linha desejada. No momento em que a linha for selecionada, detalhes do pacote serão exibidos nas duas outras partes.

A parte intermediária disponibiliza informações detalhadas do pacote e o quadro que foram capturados e por último, a parte inferior da janela apresenta as informações capturadas do pacote.

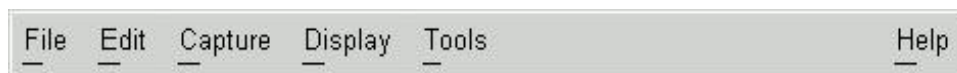
Nesta parte da janela as seguintes operações podem ser realizadas:



-  - Filter: ao clicar neste item, uma tela de construção de filtro irá aparecer.
-  - Caixa de Texto: aqui o usuário do programa pode digitar uma expressão de filtro de exibição que retornará apenas os pacotes desejados
-  - Botões reset e apply: removem ou adicionam o filtro desejado
-  - nesta parte são exibidas mensagens de aviso.

3.3.1 Barra de Menu do Ethereal

A figura abaixo representa a barra de menu do Ethereal. A barra de menu é composta pelos menus: *File*, *edit*, *capture*, *display*, *tools* e *help*.



Barra de Menu

- *Menu File*: contém as opções de abrir, salvar, imprimir os arquivos de captura, imprimir dados de pacotes e a opção de fechar o programa.
- *Menu Edit*: contém itens de procura, ir para um quadro, marcar um ou mais quadros, editar as preferências, criar filtros, ativar ou desativar dissecadores de protocolos.
- *Menu Capture*: permite iniciar e parar a captura de informações.
- *Menu Display*: contém opções para modificar informações de exibição, exibir quadros que se enquadram em uma condição, colocar cores diferentes às informações capturadas, entre outras.

- *Menu Tools*: contém itens para exibir os plugins instalados, seguir e remontar um fluxo TCP, obter um resumo dos pacotes capturados e exibir informações estatísticas.
- *Menu Help*: contém informações básicas de ajuda e informações do software.

3.3.2 Capturando Informações com o Ethereal

Ao clicar na opção *Start* do menu *Capture* aparecerá uma janela, conforme a que esta exposta abaixo. Nesta janela o usuário pode configurar as opções de captura das informações desejadas.

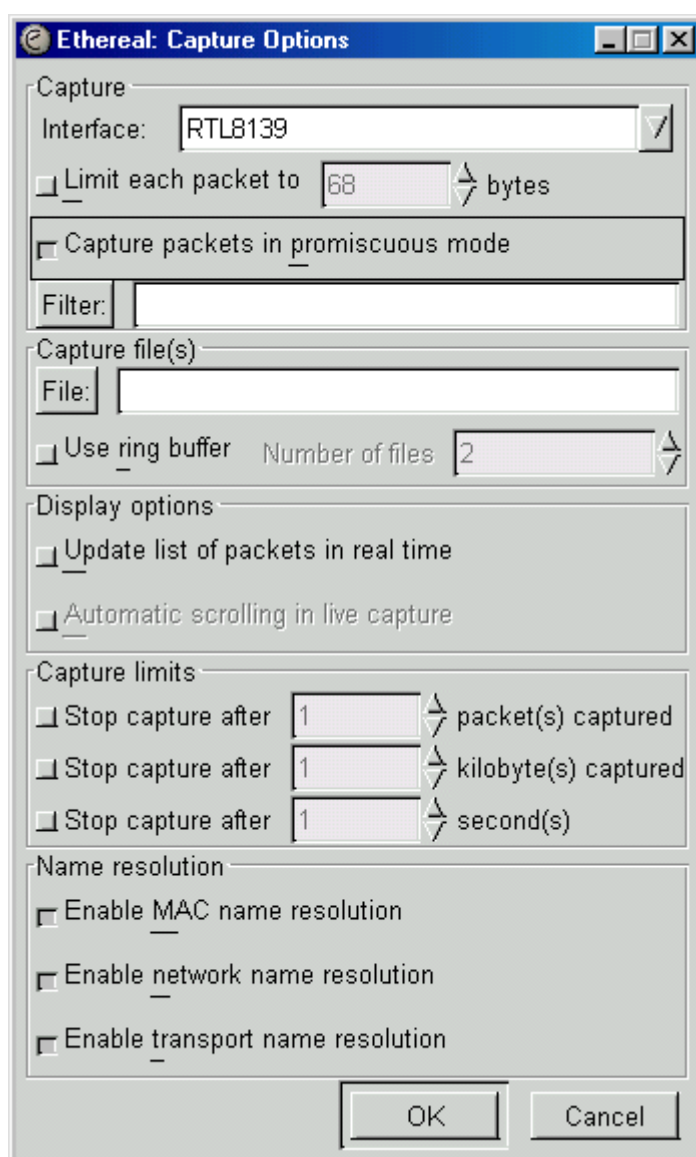


Figura 02 – Janela Configurações de Captura

A janela é composta por painéis (*capture*, *capture files*, *display options*, *capture limits* e *name resolution*). A seguir estão descritas as principais opções constantes em cada um dos painéis da janela de opções.

Painel Capture

- *Interface*: nesta caixa de seleção, o programa lista as interfaces existentes na máquina, onde o usuário poderá escolher uma das interfaces na qual deseja capturar as informações.
- *Limit each packet to*: quando selecionada esta opção permite determinar o tamanho de cada pacote em bytes.
- *Capture packets in promiscuous mode*: quando selecionada esta opção, o software irá capturar todos os quadros que a interface de rede escutar, inclusive os quadros que não são destinados a ela.
- *Filter*: nesta caixa, o usuário pode incluir uma expressão de filtro para realizar a captura.

Painel Capture File(s)

- *File*: nesta caixa o usuário indica o arquivo que será utilizado para salvar as informações capturadas.

Painel Display Options

- *Update list of packets in real time*: atualiza a lista de pacotes em tempo real.

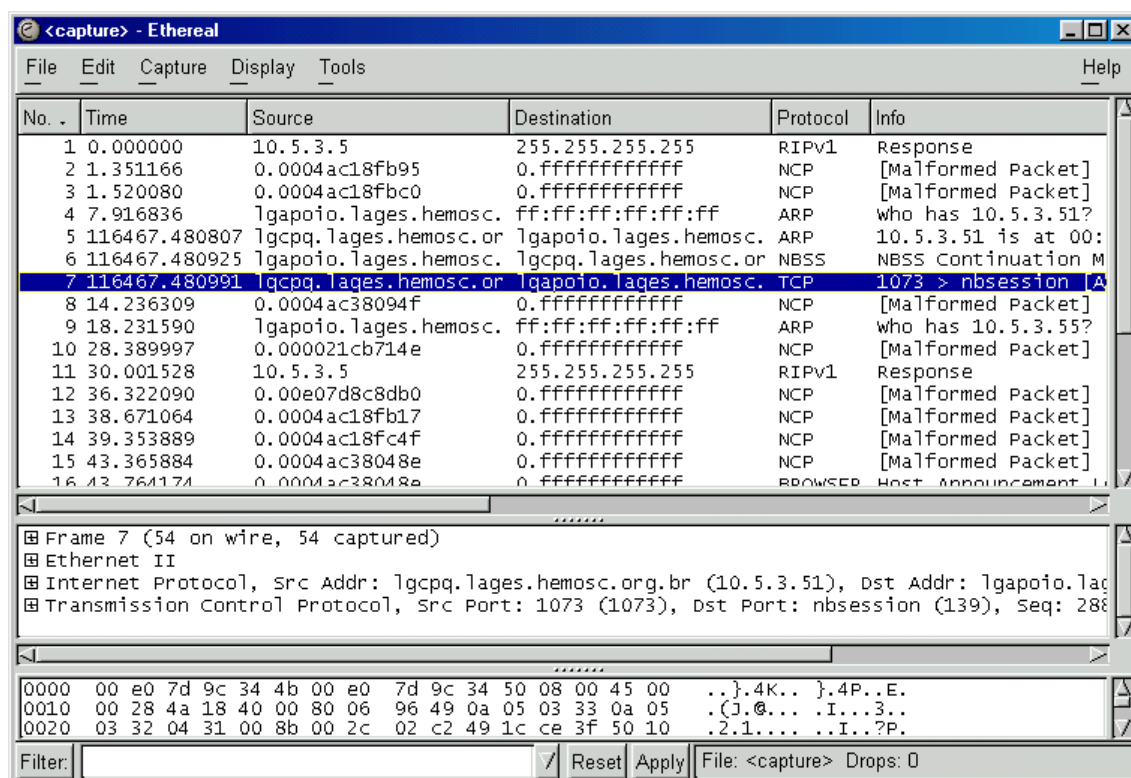
Painel Name Resolution

- *Enable network name resolution*: quando selecionada esta opção, o software fará a tradução dos endereços IP em nomes para facilitar a compreensão dos dados capturados.
- *Enable transport name resolution*: quando selecionada esta opção, o software irá converter os números de portas conhecidas em serviços.

Após selecionar as opções desejadas, finalmente clicar o botão OK para que o Ethereal possa realizar a captura das informações.

3.3.3 Visualizando as Informações Capturadas

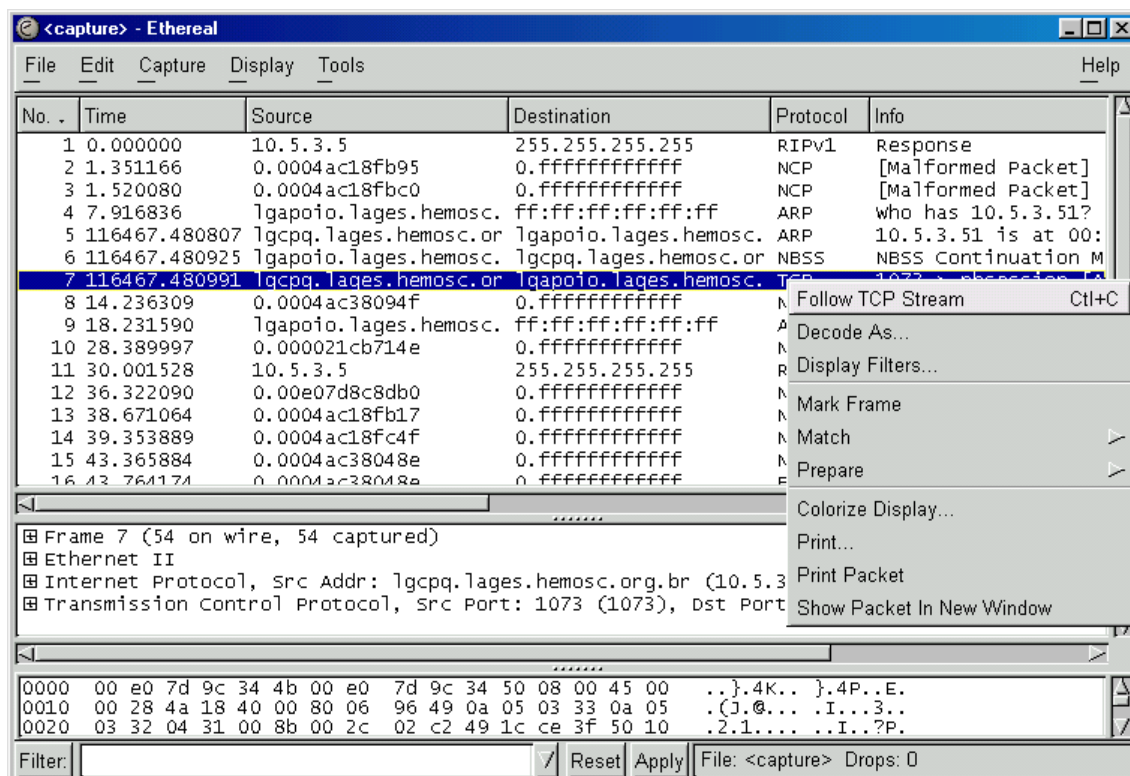
Assim que as informações são capturadas ou quando é aberto um arquivo de captura, o software exibe uma janela contendo detalhes dos pacotes que foram capturados, tais como: ordem de captura, tempo, endereço do destinatário, tipo de protocolo, informações etc. Para visualizar os pacotes capturados, analisar a figura abaixo:



Janela 03 – Informações capturadas

Observar que os pacotes são listados na ordem em que são capturados. Para obter informações de um determinado pacote, clicar com o mouse na linha da lista que contém o pacote desejado. Assim que uma linha é selecionada, o software exibe na parte intermediária da janela informações em forma de árvores do pacote correspondente. Para obter maiores informações do pacote clicar sobre o sinal + do item desejado.

Quando o usuário clicar com o botão direito do mouse sobre alguma linha da lista de pacotes, aparecerá um menu rápido, conforme o que segue:



Janela 04 – Menu Rápido

O menu rápido é composto por uma série de opções, dentre as quais selecionou-se as opções que possuem maior utilidade:

Menu Rápido

- *Follow TCP Stream*: esta opção permite ao usuário visualizar todos os dados do fluxo entre os dois pontos da conexão TCP. O Ethereal irá configurar automaticamente a expressão de filtro de exibição para listar apenas as informações do fluxo examinado e que será remontado.
- *Mark Frame*: esta opção permite ao usuário marcar os quadros que deseja e imprimir e ou salvar apenas os quadros que foram selecionados.
- *Show Packet in New Window*: esta opção permite ao usuário visualizar as informações detalhadas do pacote em uma nova janela.
- *Print e Print Packet*: esta opção permite ao usuário imprimir informações resumidas ou detalhadas de um pacote. Além disso, esta opção possibilita direcionar a impressão para um arquivo, para utilizá-lo posteriormente.

3.3.4 Aplicando Filtros na Exibição de Pacotes

Para realizar a filtragem dos pacotes, o analisador de rede Ethereal faz uso de duas linguagens de filtro. Uma é utilizada na captura e a outra é aproveitada para a exibição dos pacotes.

Os filtros de exibição permitem que o usuário se atenha apenas nos pacotes que deseja analisar. Através dos filtros, vários critérios de seleção podem ser utilizados, tais como: selecionar os pacotes de um determinado protocolo; presença de um determinado campo; valores em algum campo dos pacotes; realizar comparações entre campos.

Para selecionar um protocolo de determinado tipo, basta digitar a expressão de filtro no campo **Filter** localizado na parte inferior da tela principal do Ethereal, conforme descrito anteriormente. Todas as expressões de filtros devem ser digitadas em letras minúsculas.

É possível construir expressões de filtros, fazendo comparações e analisando valores de vários campos dos pacotes. Uma tabela foi montada, para demonstrar alguns exemplos de expressões possíveis nos filtros de exibição.

Operador C	Inglês	Exemplo de Expressão
==	eq	ip.addr == 10.0.0.5
!=	ne	ip.addr != 10.0.0.5
>	gt	frame.pkt_len > 10
<	lt	frame.pkt_len < 128
>=	ge	frame.pkt_len ge 10
<=	le	frame.pkt_len <= 100

Além dos exemplos de filtros acima, é possível definir expressões mais complexas fazendo uso de operadores lógicos **and**, **or**, **not** e **xor**. Por exemplo, podemos construir uma expressão semelhante a (ip.addr == 10.0.0.5) and (tcp.flags.syn ==1).

Esta documentação foi gerada com o intuito de ilustrar algumas das facilidades proporcionadas pelo analisador de protocolos Ethereal, porém maiores detalhes do produto estão disponíveis no site do [Ethereal].

4 Descrição do Ambiente

A rede monitorada para realizar a captura e análise dos protocolos TCP/IP foi à rede local do Hemocentro Regional de Lages – HRL. A LAN formada pelos servidores e estações da rede é uma rede Ethernet 10Mbps.

Para realizar a captura dos dados que trafegam pela rede, o software analisador de protocolos – Ethereal foi instalado em uma estação da rede do HRL. A máquina onde o analisador de rede foi instalado possui o sistema operacional Windows 98 e os seguintes recursos de hardware:

- Processador Pentium III – 750 MHz
- 128MB de memória RAM
- Disco Rígido de 20GB
- Placa de Rede Realtek 8139

Grande parte das informações capturadas foi analisada nesta mesma estação. Algumas informações capturadas foram salvas em arquivos e foram analisadas em outras estações. A figura 05 apresenta uma descrição da topologia da rede onde ocorreu a captura e análise dos dados.

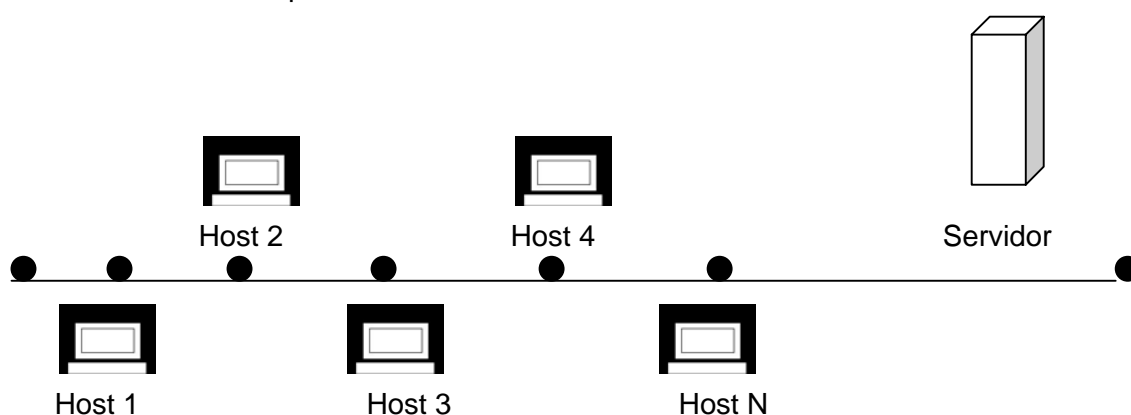


Figura 05 – Descrição Ambiente da rede do HRL

5 Análise dos Protocolos

O objetivo principal do trabalho é, através de um software analisador de protocolos, monitorar o tráfego da rede do HRL observando sobre tudo o comportamento dos protocolos TCP/IP. A medida em que os pacotes foram sendo capturados, freqüentemente surgiram dúvidas e para solucionar o problema realizou-se um estudo do funcionamento dos protocolos. Neste sentido, alguns conceitos serão descritos para auxiliar na interpretação das informações capturadas.

5.1 Protocolo IP

O protocolo IP cria uma visão virtual da rede, pois oculta a rede física subjacente. É um protocolo não confiável, de melhor esforço e sem conexão. O protocolo IP não garante a entrega dos pacotes. Deste modo os fragmentos individuais enviados pelo IP podem ser perdidos ou chegar fora de ordem. Mais importante, se uma origem envia múltiplos datagramas para um determinado destino, os fragmentos dos datagramas podem chegar em ordem arbitrária.

5.1.1 Datagrama IP

A unidade de transferência de dados do protocolo IP chama-se datagrama IP. Ele é formado por um cabeçalho contendo informações do IP e dados que são relevantes apenas para os protocolos de nível mais alto. O cabeçalho de um datagrama IP tem no mínimo 20 bytes de comprimento. Na figura abaixo podemos ver o formato de um datagrama IP.

Vers	HLEN	Type of Service	Total Length	
Identification			Flags	Fragment Offset
Thyme to Live	Protocol		Header Checksum	
Source IP Address				
Destination IP Address				
Options			Padding	

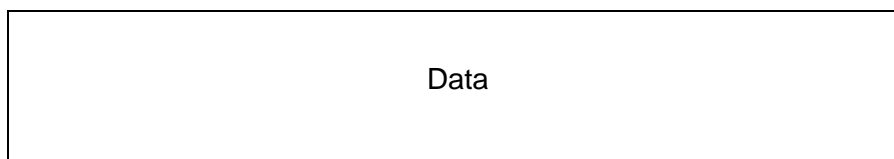


Figura 06 - Formato de um datagrama IP

5.1.1.1 Campos de um Datagrama IP

Vers: campo que contém a versão do protocolo IP. Na rede em estudo, assim como a maioria das redes IP existentes atualmente, a versão do protocolo utilizado é 4.

HLEN: campo que armazena o tamanho do cabeçalho do datagrama IP.

Type of Service: campo que indica a qualidade de serviço requerida pelo datagrama IP. O campo tipo de serviço possui o tamanho de um byte e está dividido em três partes: {Prioridade, TOS, MBZ}

Total Length: campo que contém o comprimento total do datagrama. Este é especificado em bytes e inclui tudo o que há no datagrama: cabeçalho e dados.

Identification: campo que permite que o host de destino determine a qual datagrama pertence um fragmento recém-chegado. O *identification* é o número que identifica o datagrama.

Flags: campo que armazena *flags* de controle. O primeiro bit deste campo é reservado para uso futuro. O segundo bit é usado como um sinalizador de fragmentação, quando este valor for 1 a fragmentação do pacote não é permitida. O terceiro bit sinaliza se existem mais fragmentos ou se o fragmento recebido é o último de um conjunto de fragmentos. Quando o valor deste bit é 0 indica que este é o último fragmento de um datagrama.

Fragment Offset: campo utilizado com datagramas fragmentados para ajudar no reagrupamento completo do datagrama. O valor é o número de partes de 64 bits, sendo que os bytes do cabeçalho não são contados, que estão contidos em fragmentos anteriores. No primeiro ou único fragmento, este valor é sempre zero.

Time to Live: campo que funciona como um contador para controlar em segundos a vida útil do pacote. Cada roteador por onde o datagrama passa subtrai deste campo o tempo necessário para o seu processamento. O roteador processa um datagrama em menos de um segundo, sendo assim ele subtrai o valor 1 deste campo, assim o TTL torna-se uma métrica do número de saltos. Quando o valor deste campo chegar a zero ele é descartado e um pacote de advertência é enviado para o host de origem. Este

recurso evita que os datagramas fiquem vagando indefinidamente, algo que aconteceria se as tabelas de roteamento fossem danificadas.

Protocol: campo que especifica o protocolo de nível superior (TCP ou UDP) para o qual o IP deve entregar este datagrama.

Header Checksum: campo que confere através de uma soma, apenas o cabeçalho do datagrama. Essa soma de verificação é útil para a detecção de erros gerados por palavras de memória danificadas em um roteador. Se o checksum não for igual ao conteúdo o datagrama é descartado, porque pelo menos um bit do cabeçalho está danificado e o datagrama pode até mesmo ter chegado no destino incorreto.

Source IP Address: campo que contém o endereço do host que envia o datagrama.

Destination IP Address: campo que contém o endereço do host que deve receber o datagrama.

Options: campo projetado para permitir que versões posteriores do protocolo incluam informações inexistentes no projeto original, possibilitando a experimentação de novas idéias e evitando a alocação de bits de cabeçalho para informações raramente necessárias, existem opções de tamanho variável.

Padding: campo que controla a utilização das opções. Se uma opção for utilizada o datagrama é preenchido com zeros até que se torne múltiplo de 32 bits.

Data: campo que contém os dados efetivamente transportados.

As informações contidas na da figura abaixo foram capturados no Ethereal e demonstram todos os campos de um datagrama IP.

```
Internet Protocol
  Version: 4
  Header length: 20 bytes
  Type of service: 0x00 (None)
    000. .... = Precedence: routine (0)
    ...0 .... = Delay: Normal
    .... 0... = Throughput: Normal
    .... .0.. = Reliability: Normal
    .... ..0. = Cost: Normal
  Total Length: 48
  Identification: 0xbc18
  Flags: 0x04
    .1.. = Don't fragment: set
    ..0. = More fragments: not set
  Fragment offset: 0
  Time to live: 128
  Protocol: TCP (0x06)
  Header checksum: 0x264b (correct)
  Source: lgcpg.lages.hemosc.org.br (10.5.3.51)
  Destination: NETHEMO (10.5.1.40)
  Data
```

Figura 07 – Campos de um datagrama IP

5.1.2 Fragmentação IP

Sempre que um datagrama IP percorre por vários hosts, pode ocorrer de o mesmo se defrontar com diferentes redes físicas. Cada tecnologia de hardware especifica a quantidade máxima de dados que um quadro pode transportar. O limite é conhecido como unidade máxima de transmissão – MTU. Não existe nenhuma exceção ao limite de MTU, o hardware de rede não é projetado para aceitar ou transferir quadros que transportem mais dados do que o MTU permite. Deste modo, um datagrama deve ser menor ou igual ao MTU da rede ou ele não poderá ser encapsulado para transmissão.

Como um datagrama não pode ser maior do que o MTU de uma rede através do qual é enviado, definiu-se um esquema para fragmentar os datagramas longos, e recupera-los no host de destino.

Quando um roteador recebe um datagrama que é maior do que o MTU da rede pelo qual deve ser enviado, o roteador divide o datagrama em pequenos pedaços chamados de fragmentos. Cada fragmento usa o formato de datagrama IP, mas carrega apenas parte dos dados. O tamanho mínimo do quadro deve ser de 68 bytes. Assim se uma rede oferecer um tamanho menor que este, a fragmentação e o reagrupamento fica sob responsabilidade das camadas inferiores e de maneira transparente ao IP.

No Ethereal é possível identificar se um datagrama foi ou não fragmentado, conforme ilustra a figura 08. Neste caso, tem-se um datagrama não fragmentado, onde o *flag more fragments* e *fragment offset* possuem valor zero.

```
Internet Protocol
  Version: 4
  Header length: 20 bytes
  Type of service: 0x00 (None)
  Total Length: 50
  Identification: 0xaa5b
  Flags: 0x04
    .1.. = Don't fragment: Set
    ..0. = More fragments: Not set
  Fragment offset: 0
  Time to live: 253
  Protocol: TCP (0x06)
  Header checksum: 0x656b (correct)
  Source: NETHEMO (10.5.1.40)
  Destination: lgcpq.lages.hemosc.org.br (10.5.3.51)
```

Figura 08 - Cabeçalho de um pacote IP não fragmentado

Em uma inter-rede que contém redes heterogêneas, as restrições de MTU podem causar um problema. Em particular, como um roteador pode conectar redes com valores de MTU diferentes, o roteador pode receber de uma rede um datagrama que não pode ser enviado para outra.

Quando um datagrama for maior que o MTU da rede da qual ele deve ser enviado, o roteador realiza a fragmentação para resolver o problema de MTUs heterogêneas, dividindo o datagrama em fragmentos.

Analisando os pacotes com Ethereal, observou-se que um fragmento tem o mesmo formato que outros datagramas – um bit no campo *Flags* do cabeçalho que indica se um datagrama é um fragmento ou um datagrama completo. Outros campos do cabeçalho são informações atribuídas que podem ser usadas para remontar os fragmentos de forma a reproduzir o datagrama original. Em particular o campo *Fragment Offset* no cabeçalho de um fragmento especifica a que lugar, o datagrama original, o fragmento pertence.

Visando a fragmentar um datagrama para transmissão através de uma rede, um roteador usa o MTU da rede e o tamanho do cabeçalho de datagrama para calcular a quantidade máxima de dados que podem ser enviadas em cada fragmento e o número de fragmentos que serão necessários. O roteador então cria os fragmentos. Primeiramente o roteador inicia cada fragmento com uma cópia do cabeçalho original e então modifica os campos do cabeçalho individual. Por exemplo, o roteador configura o bit apropriado no campo *Flags* para indicar que o datagrama é um fragmento. Finalmente, o roteador copia os dados apropriados do datagrama original no fragmento e transmite o resultado. A figura 09 demonstra os fragmentos de um datagrama recebido pela máquina **lgcpq**.

```
Internet Protocol
  Version: 4
  Header length: 20 bytes
  Type of service: 0x00 (None)
  Total Length: 1500
  Identification: 0xbd9f
  Flags: 0x02
    .0.. = Don't fragment: Not set
    ..1. = More fragments: Set
  Fragment offset: 0
  Time to live: 121
  Protocol: UDP (0x11)
  Header checksum: 0xd6bc (correct)
  Source: NETHEMO (10.5.1.40)
  Destination: lgcpq.lages.hemosc.org.br (10.5.3.51)
```

Figura 09 - Cabeçalho do primeiro fragmento IP recebido por lgcpq

Quando estes fragmentos chegam até o host destino, os dados precisam ser reagrupados em um único datagrama. Observe que o host emissor escolheu um número único para identificação do datagrama. Como a fragmentação não altera este campo, os fragmentos que chegam no receptor são identificados através do número existente no campo identificação juntamente com o endereço IP de origem e de destino. Além disso, o fragmento que carrega o pedaço final dos dados tem um bit adicional ligado no cabeçalho.

```
Internet Protocol
Version: 4
Header length: 20 bytes
Type of service: 0x00 (None)
Total Length: 1500
Identification: 0xbd9f
Flags: 0x02
    .0.. = Don't fragment: Not set
    ..1. = More fragments: Set
Fragment offset: 1480
Time to live: 121
Protocol: UDP (0x11)
Header checksum: 0xd5d3 (correct)
Source: NETHEMO (10.5.1.40)
Destination: lgcpq.lages.hemosc.org.br (10.5.3.51)
```

Figura 10 - Cabeçalho do segundo fragmento IP recebido por lgcpq

Para remontar os fragmentos que chegam, o host receptor reserva um buffer na memória assim que o primeiro fragmento chega. É então iniciada uma rotina que marca o tempo. Um receptor não pode armazenar fragmentos por um tempo arbitrariamente longo porque os fragmentos ocupam espaço na memória do receptor. Para evitar exaurir a memória, o IP especifica um tempo máximo para armazenar fragmentos. Quando um primeiro fragmento de um determinado datagrama chega, o receptor inicia um temporizador. Se todos os fragmentos de um determinado datagrama chegam antes do temporizador expirar, o receptor cancela o temporizador e remonta o datagrama. Porém se o temporizador expira antes de todos os fragmentos chegarem, o host receptor descarta aqueles fragmentos que chegaram.

5.1.3 Resolução de Endereços

Para garantir que todos os computadores concordem no formato e significado exato de mensagens usadas para resolver endereços, o suíte de protocolos TCP/IP inclui um Protocolo de Resolução de Endereço – ARP. O protocolo

ARP é responsável pela resolução de endereços IP de alto nível em endereços físicos de rede – MAC.

Durante a fase de análise dos protocolos, para resolver o processo de tradução ARP, os seguintes passos foram observados:

- O protocolo IP solicita ao ARP a conversão de um endereço IP para endereço MAC;
- O protocolo ARP envia uma mensagem do tipo broadcast, requisitando o endereço MAC (*arp request*);
- Todas as máquinas da rede recebem essa mensagem e realizam uma comparação com o endereço IP solicitado e com o próprio endereço IP;
- A máquina cujo endereço IP for igual ao solicitado enviará uma mensagem de resposta (*ARP reply*)
- O protocolo ARP recebe a mensagem, contendo o endereço físico da máquina configurada com o endereço IP solicitado, que será utilizado para a transmissão do pacote.

Na figura 11 e 12 tem-se respectivamente, uma solicitação e uma resposta ARP de duas máquinas do HRL utilizadas na experimentação. A figura 11 demonstra uma solicitação ARP enviada pela máquina **lgcpq** com a finalidade de conhecer o endereço físico da máquina **netadmin**.

```
Address Resolution Protocol (request)
  Hardware type: Ethernet (0x0001)
  Protocol type: IP (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (0x0001)
  Sender MAC address: 00:e0:7d:9c:34:50 (lgcpq.lages.hemosc.org.br)
  Sender IP address: lgcpq.lages.hemosc.org.br (10.5.3.51)
  Target MAC address: 00:00:00:00:00:00 (NETADMIN)
  Target IP address: NETADMIN (10.5.3.2)
```

Figura 11 - Solicitação ARP enviada pela máquina lgcpq

```
Address Resolution Protocol (reply)
  Hardware type: Ethernet (0x0001)
  Protocol type: IP (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (0x0002)
  Sender MAC address: 00:60:94:a3:11:1c (NETADMIN)
  Sender IP address: NETADMIN(10.5.3.2)
  Target MAC address: 00:e0:7d:9c:34:50 (lgcpq.lages.hemosc.org.br)
  Target IP address: 150.162.63.10 (150.162.63.10)
```

Figura 12 - Resposta ARP enviada pela máquina netadmin

O processo descrito anteriormente, por ser baseado em mensagens *broadcast*, pode ter impacto no desempenho da rede. Para evitar isso, as máquinas que originam uma requisição ARP guardam o resultado numa tabela em memória, chamada cache ARP. Uma vez sabendo o endereço físico da máquina **netadmin**, **lgcpq** manterá a vinculação de endereços em cache. Na figura abaixo podemos ver o conteúdo do cache da máquina **lgcpq** após a resposta de **netadmin**.

Interface: 150.162.63.10 on Interface 0x1000002		
Endereço Internet	Endereço físico	Tipo
10.5.3.2	00-60-94-a3-11-1c	dinâmico
10.5.3.53	00-04-10-99-2d-13	dinâmico

Figura 13 - Conteúdo do cache ARP da máquina lgcpq

5.2 Protocolo TCP

O TCP é o protocolo mais complexo do conjunto de protocolos da Internet, pois executa uma tarefa aparentemente impossível: usa o serviço de datagramas não-confiável oferecidos pelo IP ao enviar dados para outro computador, mas fornece um serviço confiável de entrega de dados para programas aplicativos.

5.2.1 Formato do Segmento TCP

As entidades TCP transmissoras e receptoras trocam dados na forma de segmentos. Os segmentos são trocados para estabelecer conexões, transferir dados, enviar reconhecimentos e finalizar conexões. Cada segmento possui um cabeçalho que transporta a identificação esperada e as informações de controle. A figura 14 apresenta o formato de um segmento TCP.

O TCP utiliza a técnica de *Piggybacking*, um reconhecimento que vai de uma host A para B pode ser colocado em um mesmo segmento de dados que está sendo enviado de A para B, embora o reconhecimento refere-se a dados enviados da máquina B para A.

Source Port			Destination Port		
Sequence Number					
Acknowledgement Number					
Offset	Reserved	Flags	Window		
Checksum			Urgent Pointer		
Options					Padding
Data					

Figura 14 - Formato de um segmento TCP

5.2.1.1 Campos de um Segmento TCP

Source Port e Destination Port: campos no cabeçalho TCP que contêm os números de portas TCP que identificam os programas de aplicação dos extremos de uma conexão.

Sequence Number: campo que contém o número de seqüência referente ao fluxo de dados que caminha na mesma direção do segmento. Identifica a posição no fluxo de bytes do segmento enviado pelo transmissor.

Acknowledgement Number: campo que identifica o número de reconhecimento do dado recebido e a referência para o próximo. Esse campo refere-se ao fluxo de dados na direção contrária ao segmento.

Offset: campo que contém um inteiro que especifica o início da porção de dados do segmento. Este campo é necessário já que o campo *options* tem tamanho variável dependendo de quais opções tenham sido incluídas. De modo que o tamanho do cabeçalho TCP varia dependendo das opções selecionadas.

RES: campo reservado para uso futuro.

CODE: campo de 6 bits que determina o propósito e conteúdo do segmento e é codificado da seguinte maneira, iniciando com os bits da esquerda para a direita:

URG: o valor 1 é atribuído a *URG* se *Urgent Pointer* estiver sendo usado. Urgente pointer é usado para indicar um deslocamento de bit no número de seqüência no qual os dados urgentes deverão estar.

ACK: o valor 1 é atribuído a *ACK* para indicar que *Acknowledgement number* é válido.

PSH: indica dados com o flag Push.

RST: utilizado para reinicializar uma conexão.

SYN: utilizado para estabelecer conexões.

FIN: utilizado para finalizar uma conexão.

Window: campo que indica quantos bytes o TCP pode receber em seu buffer.

Checksum: campo utilizado para verificar a integridade do cabeçalho e dos dados do segmento TCP.

Urgent Pointer: campo que utilizado para informar que alguns dados são urgentes. Tais dados serão expedidos tão breves quanto for possível.

Options: campo utilizado para estabelecer parâmetros com o outro lado da conexão.

Padding: este campo é usado para assegurar de que o cabeçalho tenha um tamanho que seja múltiplo de 32 bits.

```

Source port: 80 (80)
Destination port: 1688 (1688)
Sequence number: 10072771
Next sequence number: 1501232104
Acknowledgement number: 1501232104
Header length: 20 bytes
Flags: 0x0018 (PSH, ACK)
  0... .. = Congestion Window Reduced (CWR): Not set
  .0... .. = ECN-Echo: Not set
  ..0. .... = Urgent: Not set
  ...1 .... = Acknowledgment: Set
  .... 1... = Push: Set
  .... .0.. = Reset: Not set
  .... ..0. = Syn: Not set
  .... ...0 = Fin: Not set
Window size: 8541
Checksum: 0xea8b (correct)
Data

```

Figura 15 - Formato de um segmento TCP capturado

5.2.2 Estabelecimento de Conexões TCP

Muitas aplicações na arquitetura Internet fazem uso de serviços orientado a conexão, os quais são proporcionados pelo TCP. Quando estas aplicações fazem uso da rede, uma conexão entre o ponto emissor e o ponto receptor precisa ser estabelecida.

Um pedido de abertura de conexão pode ser efetuado através de dois métodos: passivo ou ativo. No modo de abertura passiva, o processo usuário fica em estado de espera aguardando a chegada pela rede de um pedido de início de conexão. Na abertura ativa, o usuário precisa estabelecer uma conexão com um processo servidor remoto, ele cria então um processo cliente para iniciar o circuito e faz um pedido de abertura ativa. Neste caso o processo na outra extremidade já deve ter feito a abertura passiva e estar esperando pelas chamadas de entrada.

Para evitar a ocorrência de problemas como a duplicação da mensagem ou a perda da mensagem de confirmação do estabelecimento da conexão, o processo de estabelecimento de conexão pelo TCP ocorre em três etapas. Através do software analisador Ethereal é possível verificar como ocorre o estabelecimento de tais conexões. A seguir será descrito o estabelecimento de uma conexão TCP entre duas máquinas do HRL. A primeira é a **lgcpq** com o endereço IP 10.5.3.51 e a segunda é a **netadmin** com endereço IP 10.5.3.53.

Inicialmente **lgcpq** envia o número de seqüência para **netadmin** e configura o bit SYN para 1, informando que os números de seqüência devem ser

sincronizados. Conforme descrito na figura abaixo, o número de seqüência é **10072474**.

```

Transmission Control Protocol
Source port: 1688 (1688)
Destination port: 80 (80)
Sequence number: 10072474
Header length: 28 bytes
Flags: 0x0002 (SYN)
  0... .. = Congestion Window Reduced (CWR): Not set
  .0.. .. = ECN-Echo: Not set
  ..0. .. = Urgent: Not set
  ...0 .. = Acknowledgment: Not set
  .... 0.. = Push: Not set
  .... .0.. = Reset: Not set
  .... ..1. = Syn: Set
  .... ...0 = Fin: Not set
Window size: 8192
Checksum: 0x8709 (correct)

```

Figura 16 - Primeiro segmento do estabelecimento da conexão

A máquina **lgcpq** pode informar o tamanho máximo do segmento, O valor informado indica a quantidade de dados que podem ser inseridos após o cabeçalho TCP. Em uma rede Ethernet, como é o caso da rede que está sendo monitorada, este valor é calculado partindo do princípio que:

- 1460 Bytes são reservados para o campo de dados do TCP
- 20 Bytes é, normalmente, o tamanho do cabeçalho TCP
- 20 Bytes é, normalmente, o tamanho do cabeçalho IP
- 14 Bytes é o tamanho do header Ethernet
- 4 Bytes é o tamanho do CRC Ethernet
- 1518 Bytes que é o tamanho máximo de quadro Ethernet

Na segunda etapa da conexão, a máquina **netadmin** ao processar o pacote contendo o segmento enviado por lgcpq, envia um novo pacote contendo um segmento TCP com o seu número de seqüência, com os flags SYN e ACK configurados para 1 e confirmando o próximo número de seqüência a ser recebido (**10072474 + 1 = 10072475**).

```

Transmission Control Protocol
Source port: 80 (80)
Destination port: 1688 (1688)
Sequence number: 1501231885
Acknowledgement number: 10072475
Header length: 28 bytes
Flags: 0x0012 (SYN, ACK)
  0... .... = Congestion Window Reduced (CWR): Not set
  .0... .... = ECN-Echo: Not set
  ..0. .... = Urgent: Not set
  ...1 .... = Acknowledgment: Set
  .... 0... = Push: Not set
  .... .0.. = Reset: Not set
  .... ..1. = Syn: Set
  .... ...0 = Fin: Not set
Window size: 8760
Checksum: 0xfc1f (correct)

```

Figura 17 - Segundo segmento do estabelecimento da conexão

Na última etapa a estação **lgcpq**, ao receber o pacote contendo o segmento TCP enviado por **netadmin**, envia um novo pacote com um segmento TCP contendo o flag ACK configurado para 1 e confirmando o próximo número de seqüência que espera receber ($1501231885 + 1 = 1501231886$). A partir de então, a máquina **lgcpq**, modifica o seu estado para "conexão estabelecida", estabelecendo assim a conexão.

```

Transmission Control Protocol
Source port: 1688 (1688)
Destination port: 80 (80)
Sequence number: 10072475
Acknowledgement number: 1501231886
Header length: 20 bytes
Flags: 0x0010 (ACK)
  0... .... = Congestion Window Reduced (CWR): Not set
  .0... .... = ECN-Echo: Not set
  ..0. .... = Urgent: Not set
  ...1 .... = Acknowledgment: Set
  .... 0... = Push: Not set
  .... .0.. = Reset: Not set
  .... ..0. = Syn: Not set
  .... ...0 = Fin: Not set
Window size: 6432
Checksum: 0x64ed (correct)

```

Figura 18 - Terceiro segmento do estabelecimento da conexão

A máquina **netadmin**, ao receber o pacote com o segmento TCP enviado por **lgcpq**, modifica o seu estado para "conexão estabelecida", estabelecendo assim a conexão.

A importância de conhecer como as conexões TCP são estabelecidas é que com isso podemos analisar algumas características importantes do tráfego da rede. Através de filtros, podemos exibir apenas os pacotes que contêm segmentos TCP que fazem parte do estabelecimento de conexões e com isso ter uma idéia de quantas conexões uma determinada estação recebeu em um determinado momento.

Na seqüência, para demonstrar o que foi relatado no parágrafo anterior, realizou-se uma análise para verificar o número de segmentos que chegam na estação **netadmin** com o flag **SYN** configurado para 1. Com posse destas informações, podemos saber quantas solicitações de início de conexões chegaram até a máquina **lgadmini**. A Figura 19 exibe um resumo de algumas conexões capturadas num curto espaço de tempo.

TCP	42395	>	80	[SYN]	Seq=888608419	Ack=0	Win=5840	Len=0
TCP	42394	>	80	[SYN]	Seq=905985288	Ack=0	Win=5840	Len=0
TCP	1932	>	80	[SYN]	Seq=2518016466	Ack=0	Win=64240	Len=0
TCP	1933	>	80	[SYN]	Seq=2518122140	Ack=0	Win=64240	Len=0
TCP	42453	>	80	[SYN]	Seq=957541053	Ack=0	Win=5840	Len=0
TCP	1961	>	80	[SYN]	Seq=2536671813	Ack=0	Win=64240	Len=0
TCP	1518	>	80	[SYN]	Seq=5032667	Ack=0	Win=8192	Len=0
TCP	1519	>	80	[SYN]	Seq=5032724	Ack=0	Win=8192	Len=0
TCP	4618	>	80	[SYN]	Seq=2775600867	Ack=0	Win=64240	Len=0
TCP	4619	>	80	[SYN]	Seq=2776410247	Ack=0	Win=64240	Len=0
TCP	1032	>	80	[SYN]	Seq=478635	Ack=0	Win=8192	Len=0
TCP	1033	>	80	[SYN]	Seq=482069	Ack=0	Win=8192	Len=0

Figura 19 - Conexões TCP iniciadas com a máquina **netadmin**

Como a máquina **netadmin** possui um serviço de HTTP rodando e esperando conexões na porta 80, durante o período de análise, verificou-se que todas as solicitações de estabelecimento de conexão para esta máquina foram feitas para a referida porta. Porém nada impede que o servidor receba tentativas de conexão com portas ou serviços que não estejam disponíveis. Sendo assim, realizou-se uma captura de informações para verificar possíveis tentativas de conexão com portas ou serviços que não estão disponíveis ou que estejam bloqueadas para determinadas redes ou estações. Simulou-se uma possível tentativa de acesso à porta 23 - telnet do servidor, a máquina **netadmin**. A figura 20 apresenta as tentativas de acesso.

Protocol	Info
TCP	1090 > telnet [SYN] Seq=1711410 Ack=0 Win=819 Len=0
TCP	telnet > 1090 [RST, ACK] Seq=0 Ack=1711411 Win=0 Len=0
TCP	1090 > telnet [SYN] Seq=1711410 Ack=0 Win=8192 Len=0
TCP	telnet > 1090 [RST, ACK] Seq=0 Ack=1711411 Win=0 Len=0
TCP	1090 > telnet [SYN] Seq=1711410 Ack=0 Win=8192 Len=0
TCP	telnet > 1090 [RST, ACK] Seq=0 Ack=1711411 Win=0 Len=0
TCP	1090 > telnet [SYN] Seq=1711410 Ack=0 Win=8192 Len=0
TCP	telnet > 1090 [RST, ACK] Seq=0 Ack=1711411 Win=0 Len=0

Figura 20 - Conexões TCP iniciadas com a máquina netadmin

Verificou-se que ao receber um segmento solicitando o estabelecimento de uma conexão com uma porta de um serviço que não está disponível, a máquina **netadmin** responde com um pacote contendo um segmento que confirma o recebimento do segmento anterior, mas com o flag reset configurado para 1 e com tamanho de janela 0. Esta situação força o cancelamento da conexão. Após três tentativas, o software telnet da máquina **lgcpq** desiste de estabelecer a conexão.

5.2.3 Múltiplas Conexões em uma mesma Porta

No processo de estabelecimento de conexão o TCP especifica além do número da porta associada à aplicação, o endereço IP da máquina. A junção desses dois endereços constitui o que é chamado de soquete. O *socket* identifica uma conexão entre dois computadores, normalmente entre um cliente e um servidor. Através de sockets, aplicações em computadores distintos podem trocar informações. A interface de programação – API *sockets* define as operações que podem ser realizadas nos *sockets*

O socket é o ponto terminal de cada um dos lados do circuito lógico associado a uma conexão. Uma conexão entre dois pontos é então identificada de forma única por um par de sockets. Devido a esse arranjo, um socket de uma máquina servidora pode ser usado de forma simultânea por mais de uma conexão bastando apenas a variação do número da porta cliente destinatária, conforme demonstrado na figura abaixo:

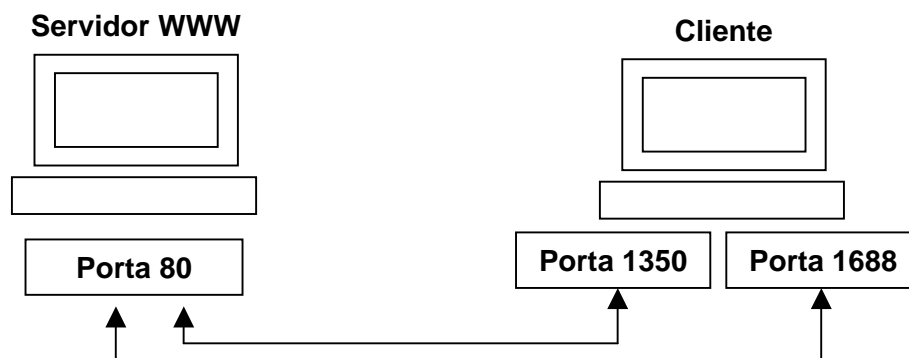


Figura 21 - Duas conexões abertas com a mesma porta do servidor.

A figura 22 exemplifica um pequeno trecho de início de duas conexões capturadas no Ethernet no servidor **netadmin** da rede do HRL. Ambas as conexões foram estabelecidas com a mesma porta do servidor.

TCP	42395	>	80	[SYN]	Seq=888608419	Ack=0	Win=5840	Len=0
TCP	42394	>	80	[SYN]	Seq=905985288	Ack=0	Win=5840	Len=0

Figura 22 - Conexões abertas com o servidor **netadmin**

5.2.4 Transferência de Dados no TCP

Assim que é estabelecida a conexão, inicia-se a transferência de dados. O TCP utiliza o recurso de *piggybacking*, o que possibilita enviar informações de reconhecimento em segmentos que estejam transportando dados para o outro lado.

As funções de seqüência e de reconhecimento garantem a ordenação dos pacotes e a identificação de pacotes que estejam faltando. Na medida que o TCP enfileira os segmentos a serem transmitidos, ele insere um número de seqüência no cabeçalho de cada segmento a ser enviado. Os dois lados da conexão utilizam estes números de seqüência para verificar a ordem correta dos segmentos.

Para confirmar o recebimento de um segmento, uma máquina envia um reconhecimento com o próximo número de seqüência esperado. Por exemplo, se a máquina **netadmin** enviar um pacote com o número de seqüência 1 para a máquina **lgcpq**, a máquina **lgcpq** reconhece o recebimento deste pacote e informa que o próximo número de seqüência esperado é o 2.

Para demonstrar a análise destes conceitos, utilizou-se uma seção de FTP estabelecida entre a máquina **lgcpq** da rede do HRL até o servidor de FTP:

ftp.cnx.com.br. Nesta conexão, transferiu-se um arquivo relftp.zip de **ftp.cnx.com.br** até a máquina **lgcpq**. A figura 23 demonstra parte da transferência.

Prot	Info
TCP	1425 > 20 [ACK] Seq=8956010 Ack=752314258 Win=8760 Len=0
TCP	20: 1460 bytes
TCP	1425 > 20 [ACK] Seq=8956010 Ack=752318638 Win=8760 Len=0
TCP	20: 1460 bytes
TCP	20: 324 bytes
TCP	1425 > 20 [ACK] Seq=8956010 Ack=752324802 Win=8760 Len=0
TCP	20: 1460 bytes
TCP	20: 1460 bytes
TCP	1425 > 20 [ACK] Seq=8956010 Ack=752327723 Win=8760 Len=0

Figura 23 - Conexão FTP demonstrando a seqüência e reconhecimento TCP

A figura 23 exibe o reconhecimento enviado por **lgcpq** para **ftp.cnx.com.br**, no qual ela confirma os dois segmentos TCP anteriores. O último ACK enviado por **lgcpq** confirmava **752324801** e esperava **752324802**. Ela recebeu dois segmentos com 1460 bytes. Portanto **752324801 + 1460 + 1460 = 75232722**. **lgcpq** envia um segmento dizendo que espera receber **752327723**.

Como não existe criptografia é possível através do analisador de protocolos observar os dados que estão trafegando na rede do HRL. Utilizando como exemplo esta conexão FTP, o software analisador, com base nos reconhecimentos e informações trocadas, é capaz de remontar a solicitação FTP efetuada. Na figura 24 tem-se a solicitação feita ao servidor FTP e que foi remontada pelo Ethereal.

No início da conexão com o servidor remoto solicitou-se o nome de usuário e uma senha, para serem utilizados na validação dos direitos de acesso. No caso da Internet, alguns sites possuem arquivos ou diretórios confidenciais, que somente podem ser acessados por usuários autorizados. Entretanto muitos servidores FTP, como é o caso, estão disponíveis para uso público e, nesse caso o nome do usuário deve ser anonymous e o e-mail como senha. Isso pode ser verificado na figura 24.


```

220-WAR FTP service
      WarFTPD 1.71.02 (Feb 14 2000) Ready
      (C)opyright 1996 - 2000 by Jarle (jgaa) Aase - all rights
reserved.
220 Please enter your user name.
USER anonymous
331 User name okay. Give your full Email address as password.
PASS madalena@hemosc.org.br
230 User logged in.
PORT 10,5,3,53,5,143
200 PORT command successful.
NLST
150 Opening ASCII mode data connection for /bin/ls (254 bytes).
226 Transfer complete. 254 bytes in 0.09 sec. (2.726 Kb/s)
TYPE I
200 Type set to I.
CWD relftp
250 "/relftp" is current directory.
TYPE A
200 Type set to A.
PORT 10,5,3,53,5,144
200 PORT command successful.
NLST
150 Opening ASCII mode data connection for /bin/ls (10 bytes).
226 Transfer complete. 10 bytes in 0.09 sec. (0.107 Kb/s)
TYPE I
200 Type set to I.
PORT 10,5,3,53,5,145
200 PORT command successful.
RETR relftp.zip
150 Opening BINARY mode data connection for relftp.zip (2578321
bytes).
426 Error. Transfere aborted. An established connection was aborted
by the software in your host machine.
QUIT
221 Goodbye. Control connection closed.

```

Figura 24 - Conexão FTP remontada pelo software analisador

5.2.4.1 Ausência de Criptografia

O projeto original da Internet, não proporcionou nenhum recurso de segurança e criptografia das informações transmitidas pelo conjunto de protocolos TCP/IP. Com o passar do tempo mecanismos de segurança e criptografia passaram a ser adicionados nos protocolos da camada de aplicação, com o objetivo de autenticar e criptografar as informações transmitidas.

Como o Ipv4 não possui nenhum mecanismo de segurança e criptografia intrínseco, utilizou o IPSec no protocolo IP no nível da camada de rede e de transporte para contornar esse problema. Embora, tais recursos tenham sido bastante utilizados

em redes privadas virtuais e no estabelecimento de túneis de criptografia a sua adoção em redes locais têm sido pequena.

Em virtude de os analisadores de protocolos serem capazes de capturar todos os protocolos da rede à medida que eles percorrem o cabeamento, qualquer pessoa pode acompanhar o seu funcionamento e visualizar os dados que trafegam na rede. Em função da ausência de segurança no protocolo IP, ao longo dos anos, vários ataques a redes e servidores foram feitos.

Para demonstrar a ausência de criptografia, na figura abaixo temos parte de um datagrama IP que foi capturado enquanto a rede estava sendo monitorada. No campo de dados deste datagrama percebe-se a presença de um **nome de usuário e senha** enviados a um servidor de correio eletrônico através do método HTTP POST.

```
POST /login.htm HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-powerpoint, application/vnd.ms-excel,
application/msword, */*
Referer: http://www.zipmail.com.br/home1.jsp
Accept-Language: pt-br
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows 98)
Host: acesso.zipmail.com.br
Content-Length: 53
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: kuv=s00910188955340375; homebol=1

upass=secret&Act_Login=+Entrar+&uname=madlages•IP<••• < < PM •
7™0•• E (•R@ ø••EEY•+-ç?
```

Figura 25 - Parte do campo de dados do segmento contendo nome do usuário e senha

5.2.4.2 Buffers, Controle de Fluxos e Janelas

O TCP utiliza um mecanismo de janela para controlar o fluxo de dados. Quando uma conexão é estabelecida, cada lado da conexão aloca um buffer para armazenar dados que chegam e envia o tamanho desse buffer para outro lado. À medida que chegam dados, o receptor envia *acknowledgments*, que especificam também o tamanho do buffer restante. A quantidade de espaço de buffer disponível a qualquer momento é chamada de janela, e uma notificação que especifica o tamanho é chamada de anúncio de janela. Um receptor envia um anúncio de janela com cada *acknowledgment*.

Se o aplicativo receptor pode ler dados tão depressa quanto eles chegam, um receptor enviará um anúncio de janela positiva junto com cada *acknowledgment*. Porém se o lado remetente opera mais rápido que o lado receptor, os dados que estão chegando irão eventualmente encher o buffer do receptor, fazendo com que o receptor anuncie uma janela de tamanho zero. Um remetente que recebe um anúncio de janela de tamanho zero deve acessar o envio até que o receptor anuncie novamente uma janela positiva.

A figura 26 apresenta parte da captura de uma transferência de arquivos efetuada pela máquina **lgcpq**. Nesta pequena parte da transmissão, através da mudança do campo Window nos segmentos TCP enviados para o servidor FTP, percebe-se a mudança no tamanho da janela de **lgcpq**.

Protocol	Info
TCP	1514 > 20 [ACK] Seq=19098175 Ack=2940213658 Win=8760 Len=0
FTP-DATA	FTP Data: 1460 bytes
TCP	1516 > 20 [ACK] Seq=19280915 Ack=2965880396 Win=6432 Len=0
TCP	1516 > 20 [ACK] Seq=19280915 Ack=2965880396 Win=8760 Len=0
FTP-DATA	FTP Data: 1460 bytes
FTP-DATA	FTP Data: 1460 bytes
TCP	1514 > 20 [ACK] Seq=19098175 Ack=2940215118 Win=8760 Len=0

Figura 26 - Tamanho da janela de **lgcpq**

5.2.5 Encerramento de Conexões TCP

Encerrar uma conexão é mais fácil do que estabelece-la. No TCP existem dois tipos de encerramento de conexão, o encerramento simétrico e o encerramento assimétrico.

No encerramento simétrico cada um dos lados é efetivamente desconectado após a entrega efetiva de todos os dados. O modo simétrico trata a conexão como duas conexões unidirecionais e exige que cada uma seja encerrada separadamente. Neste caso, o processo de desconexão é negociado, ou seja, o usuário remoto tem a opção de concordar ou não com a desconexão.

O encerramento assimétrico é abrupto e pode resultar na perda de dados. Uma conexão de TCP permite a transferência de dados full-duplex, isto significa que existem cadeias de dados independentes, fluindo em cada uma das direções da conexão. Para liberar as duas direções da conexão ambos os lados devem ser fechados, no entanto, caso desejado, pode-se encerrar só uma delas.

Para encerrar conexões, o TCP utiliza um handshake de 3 vias modificado. Para compreender o funcionamento e as informações trocadas durante um encerramento normal de uma conexão, vamos analisar o exemplo da conexão FTP descrito anteriormente entre a máquina **lgcpq** (10.5.3.51) e **ftp.cnx.com.br** (200.247.202.34).

Inicialmente, conforme apresentação da figura 27, o **ftp.cnx.com.br** envia um pacote IP com um segmento TCP contendo o sinalizador FIN configurado para 1. Neste caso, o outro lado poderá responder com uma confirmação (ACK) do FIN ou com o seu próprio FIN ou ambos. Quando um FIN chega desacompanhado, a máquina sabe que o outro lado pode ter começado a encerrar a conexão.

```

Transmission Control Protocol
Source port: 21 (21)
Destination port: 1422 (1422)
Sequence number: 562731268
Acknowledgement number: 6970125
Header length: 20 bytes
Flags: 0x0011 (FIN, ACK)
  0... .. = Congestion Window Reduced (CWR): Not set
  .0.. .. = ECN-Echo: Not set
  ..0. .... = Urgent: Not set
  ...1 .... = Acknowledgment: Set
  .... 0... = Push: Not set
  .... .0.. = Reset: Not set
  .... ..0. = Syn: Not set
  .... ...1 = Fin: Set
Window size: 18326
Checksum: 0xb310 (correct)

```

Figura 27 - Primeiro segmento TCP da desconexão, enviado por ftp.cnx.com.br

Ao receber o segmento com o sinalizador FIN, a máquina **lgcpq** envia um reconhecimento (ACK) do pedido de encerramento da conexão. Quando um FIN chega antes que o aplicativo emita um *close*, o TCP modificará o seu estado de forma que possa avisar o aplicativo de que o outro lado encerrou a conexão. Antes de emitir o último ACK da conexão, ele irá esperar o aplicativo emitir uma operação *close*. O reconhecimento enviado por **lgcpq** após receber o segmento FIN enviado pelo servidor ftp pode ser comprovado através da figura 28.

```

Transmission Control Protocol
Source port: 1422 (1422)
Destination port: 21 (21)
Sequence number: 6970125
Acknowledgement number: 562731268
Header length: 20 bytes
Flags: 0x0010 (ACK)
  0... .. = Congestion Window Reduced (CWR): Not set
  .0... .. = ECN-Echo: Not set
  ..0. .... = Urgent: Not set
  ...1 .... = Acknowledgment: Set
  .... 0... = Push: Not set
  .... .0.. = Reset: Not set
  .... ..0. = Syn: Not set
  .... ...0 = Fin: Not set
Window size: 7957
Checksum: 0xe24d (correct)

```

Figura 28 - Reconhecimento de lgcpq após receber o segmento FIN enviado pelo servidor ftp

Ao receber o ACK enviado por **lgcpq**, **ftp.cnx.com.br** irá esperar um determinado tempo pelo fechamento da conexão. Este tempo é dado porque podem existir segmentos enviados fora da ordem, então, antes de tentar enviar um segundo FIN, a estação espera um tempo.

Assim que o aplicativo da máquina **lgcpq** emitir uma operação de close, conforme figura 29, ela enviará um segmento TCP com o sinalizador FIN configurado para 1 e com o último reconhecimento enviado por ela.

```

Transmission Control Protocol
Source port: 1422 (1422)
Destination port: 21 (21)
Sequence number: 6970125
Acknowledgement number: 562731268
Header length: 20 bytes
Flags: 0x0011 (FIN, ACK)
  0... .. = Congestion Window Reduced (CWR): Not set
  .0... .. = ECN-Echo: Not set
  ..0. .... = Urgent: Not set
  ...1 .... = Acknowledgment: Set
  .... 0... = Push: Not set
  .... .0.. = Reset: Not set
  .... ..0. = Syn: Not set
  .... ...1 = Fin: Set
Window size: 7957
Checksum: 0xe24c (correct)

```

Figura 29 - FIN com o último ACK enviado por lgcpq

Ao receber o segmento FIN enviado por **lgcpq**, **ftp.cnx.com.br** envia um reconhecimento (ACK) final e encerra a conexão. Quando este último reconhecimento chega na máquina **lgcpq**, ela também encerrará a conexão.

```
Transmission Control Protocol
Source port: 21 (21)
Destination port: 1422 (1422)
Sequence number: 56731268
Acknowledgement number: 6970125
Header length: 20 bytes
Flags: 0x0010 (ACK)
  0... .. = Congestion Window Reduced (CWR): Not set
  .0... .. = ECN-Echo: Not set
  ..0. .... = Urgent: Not set
  ...1 .... = Acknowledgment: Set
  .... 0... = Push: Not set
  .... .0.. = Reset: Not set
  .... ..0. = Syn: Not set
  .... ...0 = Fin: Not set
Window size: 18326
Checksum: 0xc36f (correct)
```

Figura 30 - Último reconhecimento enviado por ftp.cnx.com.br

6 Conclusão

Até iniciar o trabalho, não existia nenhum software analisador de protocolos instalado na rede local do Hemocentro Regional de Lages - HRL. Assim que a rede começou a ser monitorada, mesmo sendo uma LAN, observou-se o quanto é importante a unidade possuir um analisador de rede. Principalmente porque no Hemocentro, trabalha-se com dados confidenciais de funcionários, pacientes e doadores de sangue. Além disso, a rede deve funcionar ininterruptamente.

Os analisadores não se enquadram inteiramente na categoria de produtos de gerenciamento de rede, mas realizam funções de controle dos dados que trafegam, gerenciam informações referentes ao cabeamento e podem ser configurados para detectar colisões.

Quando deu início na busca dos analisadores disponíveis na Web, percebeu-se que para entender o funcionamento e a utilidade de um software analisador de rede, um estudo dos protocolos deveria ser realizado.

Durante o desenvolvimento do relatório, mesmo tendo realizado um estudo dos protocolos TCP/IP dúvidas freqüentemente surgiram, o que evidencia que o apanhado do funcionamento dos protocolos não foi suficiente, contudo o relatório atendeu o objetivo previamente estabelecido: capturar e analisar os pacotes que trafegam pela rede do HRL e descrever algumas conclusões com base nas informações coletadas.

7 Referências Bibliográficas

Chappell, Laura. **Inside the TCP Handshake**. NetWare Connection, março de 2000. Disponível em <http://www.nwconnection.com/2000_03/pdf30/hand30.pdf>

Chappell, Laura. **TCP Sequencing and Sliding Windows**. NetWare Connection, maio de 2001. Disponível em <http://www.ncmag.com/2001_05/pdf51/sequence51.pdf>

Chappell, Laura. **Advanced Packet Filtering**. NetWare Connection, abril de 2001. Disponível em <http://www.ncmag.com/2001_04/pdf41/packet41.pdf>

Chappell, Laura. **Using the Sniffer Capture Window or Panel**. NetWare Connection, abril de 2001. Disponível em <http://www.nwconnection.com/2000_11/pdf11/capturen0.pdf>

Comer, Douglas E., Stevens David L. **Interligação em Rede Com TCP/IP**, Volume 2. Editora Campus, Rio de Janeiro, 1999.

Comer, Douglas E..**Redes de Computadores e Internet**, 2ª Edição. Editora Bookman, São Paulo, 2001.

Ethereal – The Ethereal Network Analyzer. Site do projeto Ethereal. Contém informações sobre o desenvolvimento do projeto, arquivos fontes, arquivos de instalação e documentação do projeto. Disponível em: <<http://www.ethereal.com>>. Acesso em: 10 de março de 2002.

Souza, Lindeberg Barros de.**Redes de Computadores**, 3ª Edição. Editora Érica, São Paulo, 2000.

Tanenbaum, Andrew S..**Redes de Computadores**, 3ª Edição. Editora Campus, Rio de Janeiro, 1997.

TCPDump Public Repository. Site do projeto TCPDump e LibPCap. Contém informações sobre desenvolvimento, arquivos de instalação, documentação, etc. Disponível em <<http://www.tcpdump.org>>. Acesso em: 02 de março de 2002.

WinPcap Brings Unix Network Tools to Windows. O autor introduz as ferramentas WinPcap, WinDump, Nmap, Ngrep e Dsniff com alguns exemplos. Disponível em: <http://security.oreilly.com/news/securingnt2_1200.html>. Acesso em: 05 de abril de 2002.

WinPcap: The Free Packet Capture Architecture for Windows. Site do projeto WinPCap. Nele podem ser obtidas informações, fontes e instalação da biblioteca. Disponível em: <<http://winpcap.polito.it>>. Acesso em: 10 de março de 2002.