



**APRIMORAMENTO DE UM SIMULADOR DIDÁTICO DE REDES DE
COMPUTADORES**

Cristiano Führt Poletti

Lajeado, junho de 2013

Cristiano Führ Poletti

APRIMORAMENTO DE UM SIMULADOR DIDÁTICO DE REDES DE COMPUTADORES

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Tecnológicas do Centro Universitário UNIVATES, como parte dos requisitos para a obtenção do título de bacharel em Engenharia da Computação.

Área de concentração: Redes de Computadores.

Orientador: Marcelo de Gomensoro Malheiros

Lajeado, junho de 2013

CRISTIANO FÜHR POLETTI

APRIMORAMENTO DE UM SIMULADOR DIDÁTICO DE REDES DE COMPUTADORES

Este trabalho foi julgado adequado para a obtenção do título de bacharel em Engenharia da Computação do CETEC e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Marcelo de Gomensoro Malheiros, UNIVATES

Mestre pela UNICAMP – Campinas, Brasil

Banca Examinadora:

Prof. Luís Antônio Schneiders, UNIVATES

Mestre pela UFRGS – Porto Alegre, Brasil

Prof. Edson Moacir Ahlert, UNIVATES

Especialista pela ULBRA – Canoas, Brasil

Coordenador do curso de Engenharia da Computação: _____

Prof. Marcelo de Gomensoro Malheiros

Lajeado, junho de 2013

RESUMO

As redes de computadores fazem parte da vida diária de empresas e de usuários domésticos, sendo uma das áreas mais importantes para a formação de profissionais de Computação. Mesmo sendo um tópico relevante, existem relativamente poucas ferramentas de apoio ao ensino de Redes de Computadores. Com este objetivo, Müller (2010) propôs o desenvolvimento de um novo simulador de redes, ainda que com algumas limitações. O presente trabalho descreve a implementação de novas funcionalidades e aperfeiçoamento de recursos já desenvolvidos anteriormente. Em particular, pretende-se expor, de forma gráfica, as etapas do processo de estabelecimento de conexões TCP e permitir que protocolos como HTTP e ICMP possam ser respondidos automaticamente, simulando o funcionamento de um servidor.

Palavras-chave: redes de computadores, simulador de redes, ferramentas de ensino

ABSTRACT

Computer networks are part of everyday life for businesses and home users, being one of the most important areas when educating Computer Science professionals. Despite being a major topic, there are relatively few tools to support the teaching of Computer Networks. With this goal, Müller (2010) proposed the development of a new network simulator, although with some limitations. This paper describes the implementation of new features and the improvement of the existing features. In particular, it is intended to expose, in graphic form, the steps made when establishing TCP connections and also allow protocols like HTTP and ICMP to be answered automatically, therefore simulating a server.

Keywords: computer networks, network simulator, teaching aids

LISTA DE FIGURAS

FIGURA 1 - PILHA DE CAMADAS DO PROTOCOLO TCP/IP	15
FIGURA 2 – QUADRO ETHERNET	17
FIGURA 3 – CABEÇALHO DO DATAGRAMA IP	19
FIGURA 4 – MENSAGEM ECHO (ICMP).....	22
FIGURA 5 – MENSAGEM DESTINATÁRIO NÃO ACESSÍVEL (ICMP).....	23
FIGURA 6 – CABEÇALHO UDP	25
FIGURA 7 – CABEÇALHO TCP.....	26
FIGURA 8 – HANDSHAKE DE ESTABELECIMENTO DA CONEXÃO TCP.....	28
FIGURA 9 – TROCA DE MENSAGENS PARA FINALIZAÇÃO DA CONEXÃO TCP.....	29
FIGURA 10 – MÁQUINA DE ESTADOS DO PROTOCOLO TCP.....	31
FIGURA 11 – REQUISIÇÃO HTTP.....	33
FIGURA 12 – RESPOSTA HTTP.....	34
FIGURA 13 – CISCO PACKET TRACER.....	36
FIGURA 14 – JANELA PARA CRIAÇÃO DE UMA MENSAGEM COMPLEXA.....	37
FIGURA 15 – LISTA DAS MENSAGENS TROCADAS.....	38
FIGURA 16 – DETALHES DAS CAMADAS DO MODELO OSI.....	39
FIGURA 17 – DETALHAMENTO DE UM PACOTE ENVIADO.....	39
FIGURA 18 – DIAGRAMA DE CLASSES DOS DATAGRAMS.....	43
FIGURA 19 – ABAS DOS PROTOCOLOS.....	44
FIGURA 20 – DIAGRAMA DE CLASSES DOS PANELS.....	45
FIGURA 21 – DIAGRAMA DE ESTADOS DO PROTOCOLO TCP.....	47
FIGURA 22 – PACOTE ICMP GERADO PELO SIMULADOR.....	50
FIGURA 23 – RODAPÉ DO SIMULADOR.....	53
FIGURA 24 – ESTADO INICIAL DA TELA PRINCIPAL.....	54
FIGURA 25 – TELA PRINCIPAL APÓS INTERAÇÕES.....	55
FIGURA 26 – TELA DE BUSCA DE NOVAS ESTAÇÕES.....	56
FIGURA 27 – TELA DE CRIAÇÃO DE PACOTES.....	57
FIGURA 28 – TELA DE VISUALIZAÇÃO DE PACOTES.....	58
FIGURA 29 – TELA TABELA ARP.....	59
FIGURA 30 – TELA NETSTAT.....	60
FIGURA 31 – MENSAGEM DE SUCESSO DO SIMULADOR.....	62

LISTA DE TABELAS

TABELA 1 – TIPOS DE MENSAGENS DO PROTOCOLO ICMP.....	21
TABELA 2 – POSSÍVEIS VALORES DE CÓDIGOS PARA UMA MENSAGEM DE DESTINATÁRIO NÃO ACESSÍVEL DO PROTOCOLO ICMP.....	23
TABELA 3 – ESTADOS DA MÁQUINA DE ESTADOS DO TCP.....	30
TABELA 4 – POSSÍVEIS CÓDIGOS DE UMA RESPOSTA HTTP.....	33
TABELA 5 – MENUS DA APLICAÇÃO.....	61

LISTA DE ABREVIATURAS

ARP:	Address Resolution Protocol
ARPANET:	Advanced Research Projects Agency Network
CRC:	Cyclic Redundancy Check
DARPA:	Defense Advanced Research Agency
DNS:	Domain Name Service
DoD:	Department of Defense
HTTP:	HyperText Transfer Protocol
ICMP:	Internet Control Message Protocol
IEEE:	Institute of Electrical and Electronics Engineers
IP:	Internet Protocol
MAC:	Media Access Protocol
MIT:	Massachusetts Institute of Technology
MTU:	Maximum Transfer Unit
OSI:	Open Systems Interconnection
OUI:	Organizationally Unique Identifier
PPP:	Point-to-Point Protocol
RFC:	Request for Comments
RTP:	Real-time Transport Protocol
TCP:	Transmission Control Protocol
UDP:	User Data Protocol

SUMÁRIO

1 INTRODUÇÃO.....	11
2 REFERENCIAL TEÓRICO.....	14
2.1 Modelo TCP/IP.....	14
2.2 Camada de Interface de redes.....	16
2.2.1 Ethernet.....	16
2.3 Camada de Rede.....	18
2.3.1 IP.....	18
2.3.2 ICMP.....	20
2.4 Camada de Transporte.....	24
2.4.1 UDP.....	24
2.4.2 TCP.....	25
2.5 Camada de Aplicação.....	32
2.5.1 HTTP.....	32
3 TRABALHOS RELACIONADOS.....	35
3.1 Simuladores comerciais e para pesquisa.....	35
3.2 Simulador Cisco Packet Tracer.....	36
4 IMPLEMENTAÇÃO.....	41
4.1 Visão geral.....	41
4.2 Estrutura das Classes.....	42
4.2.1 Classes Datagram.....	43
4.2.2 Classes Panel.....	44
4.2.3 Classe WorkStation.....	46
4.2.4 Classe Packet.....	46
4.2.5 Classe Netstat.....	46
4.2.6 Classe ArpTable.....	47
4.2.7 Classe FindStation.....	48
4.3 Modificações realizadas.....	48
4.3.1 Resposta Automática de Pacotes.....	48
4.3.2 Criação de estação secundária.....	49
4.3.3 Criação do pacote ICMP.....	49
4.3.4 Remoção da dependência da camada de aplicação.....	51
4.3.5 Criação de algoritmo de carregamento dinâmico de objetos Datagram.....	51

5 ANÁLISE DA APLICAÇÃO.....	53
5.1 Fluxo de Telas.....	53
5.1.1 Rodapé.....	53
5.1.2 Tela Principal.....	54
5.1.3 Busca de novas estações.....	56
5.1.4 Criação de pacotes.....	56
5.1.5 Visualização de pacotes.....	57
5.1.6 Tabela ARP.....	59
5.1.7 Netstat.....	59
5.1.8 Menu.....	61
5.2 Mensagens para o Usuário.....	62
5.3 Internacionalização.....	62
6 CONCLUSÃO.....	64

1 INTRODUÇÃO

Durante as primeiras décadas da existência da indústria da Informática, os sistemas computacionais eram altamente centralizados. Uma empresa de médio porte ou uma universidade contavam apenas com um ou dois computadores, enquanto as grandes instituições tinham no máximo algumas dezenas.

A fusão dos computadores e da comunicação teve uma profunda influência na forma como os sistemas computacionais são organizados. O modelo de um único computador atendendo a todas as necessidades computacionais de uma organização foi substituído por outro, em que os trabalhos são realizados por um grande número de computadores separados, porém interconectados. Esse sistema é chamado de rede de computadores.

As redes de computadores exercem um papel muito importante nas empresas, possibilitando o compartilhamento das informações do negócio. Além disso as redes auxiliam na comunicação entre os funcionários, através de e-mails ou pela telefonia IP, possibilitando também a realização do comércio eletrônico, que tem crescido rapidamente nos últimos anos.

Os usuários domésticos também utilizam largamente as redes de computadores, principalmente a Internet. A Internet possibilita acessar informações remotas, comunicar-se com outras pessoas e comprar produtos e serviços através do comércio eletrônico (TANENBAUM, 2011).

Com o aumento da complexidade dos sistemas computacionais, a demanda de velocidade, confiabilidade e escalabilidade das redes também implica em uma maior complexidade para estas. Aliado ao fato de que cada vez mais se projetam equipamentos de comunicação inteligentes, a compreensão de uma rede como um todo se torna mais difícil (MÜLLER, 2010).

Para Comer (2009), a ligação de computadores é um assunto complexo, pois existem muitas tecnologias com características distintas, que podem ser usadas para interconectar duas ou mais redes, consequentemente possibilitando muitas combinações de redes. Além disso com a falta de uma teoria de embasamento, não há uma terminologia simples e uniforme para os conceitos sobre ligação de computadores em rede.

Atualmente existem poucos recursos didáticos que auxiliem o entendimento da comunicação nas redes de computadores para pessoas com pouco conhecimento na área de redes. Existem algumas ferramentas, descritas no Capítulo 3, que simulam um ambiente de redes, porém estes sistemas são complexos e não permitem uma maior interação com os dados dos pacotes dos protocolos de rede.

Da percepção de que existe dificuldade para um professor encontrar material didático que prenda a atenção do aluno, Müller (2010) desenvolveu um simulador didático de redes na linguagem Java. Este simulador era simples e permitia, sem a necessidade de configurações avançadas, simular a criação e envio de pacotes de protocolos do modelo TCP/IP para outras estações contidas na rede.

Esse simulador apresentava ainda algumas limitações. Por exemplo, para criar um pacote era necessário escolher um protocolo para cada camada do modelo TCP/IP, porém nem todos os protocolos têm esta característica, como por exemplo o ICMP.

Outra limitação era a necessidade de conexão com outras instâncias do simulador para poder realizar a troca dos pacotes, impossibilitando que um usuário que não estivesse conectado à uma rede utilizasse a ferramenta. Em outras palavras, não era possível executar duas instâncias do simulador em um mesmo computador.

Dado este cenário, o objetivo geral do presente trabalho é dar continuidade ao simulador de redes desenvolvido por Müller (2010). Pretende-se implementar novas funcionalidades e aprimorar alguns recursos já desenvolvidos.

Como objetivos específicos pode-se citar a implementação e visualização dos estados do protocolo TCP, possibilitar múltiplos simuladores em uma mesma máquina e criar um mecanismo de autodescoberta de outros simuladores.

No Capítulo 2 deste documento será abordado o conceito de divisão de camadas de redes, dando ênfase ao modelo TCP/IP, além de detalhar os protocolos mais relevantes para o simulador de redes em questão. No Capítulo 3 são analisados alguns softwares que

desempenham tarefas semelhantes ao trabalho proposto. O Capítulo 4 descreve a implementação das novas funcionalidades, e explica a organização das principais classes desenvolvidas. No Capítulo 5 é feita uma análise da aplicação, mostrando o funcionamento das principais telas do simulador e descrevendo um experimento realizado. E finalmente, no Capítulo 6, são expostas as conclusões obtidas e indicadas possibilidades para trabalhos futuros.



2 REFERENCIAL TEÓRICO

Para Kurose (2011) a Internet é um sistema complicado possuindo muitos componentes, aplicações e protocolos, vários meios físicos diferentes para a transmissão de dados e vários tipos de sistemas finais e conexões entre eles.

Uma arquitetura de camadas permite analisar uma parcela específica e bem definida de um sistema grande e complexo. Esta divisão permite a modularização dos sistema, fazendo com que fique mais fácil modificar a implementação dos serviços prestados pelas camadas.

A ideia da divisão em camadas é fundamental porque fornece uma estrutura conceitual para o projeto de um protocolo. Em um modelo dividido em camadas, cada camada trata de um parte do problema de comunicações e geralmente corresponde a um protocolo. Os protocolos divididos em camadas são projetados de modo que a camada n de destino receba exatamente o mesmo objeto enviado pela camada n de origem (COMER, 1998).

Segundo Tanenbaum (2010), duas das mais importantes arquiteturas de rede são os modelos de referência OSI (Open Systems Interconnection) e o TCP/IP.

2.1 Modelo TCP/IP

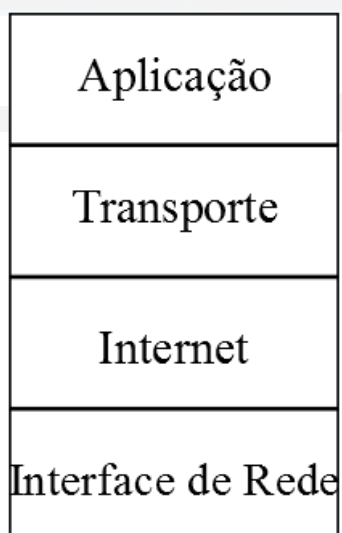
Segundo Júnior (1999), o modelo de interconexão TCP/IP foi desenvolvido pelo departamento de defesa americano para resolver o problema de interconexão de computadores heterogêneos, tendo evoluído a partir de trabalhos iniciados no MIT (Massachusetts Institute of Technology) e contando com a cooperação de várias empresas. A DARPA (Defense Advanced Research Agency), ligada ao DoD (Department of Defense), iniciou no começo dos anos 70 a busca pela interligação de diferentes computadores, possibilitando o suporte a esforços de pesquisa em laboratórios, universidades e instituições governamentais americanas.

Em 1980, a DARPA instalou os primeiros módulos TCP/IP em computadores da sua rede, a ARPANET (Advanced Research Projects Agency Network), tendo determinado que todos os computadores ligados a essa rede utilizassem os protocolos TCP/IP.

A pilha de protocolos TCP/IP representa a mais usada forma de comunicação entre computadores remotos disponibilizada atualmente, servindo de base para a Internet.

Para Comer (1998), a arquitetura TCP/IP é organizada em quatro camadas conceituais, construídas sobre uma quinta camada de hardware, como demonstrado na Figura 1.

Figura 1 - Pilha de camadas do protocolo TCP/IP



Fonte: Adaptado pelo autor com base em Comer (1998).

O objetivo de cada camada é oferecer determinados serviços às camadas superiores, isolando essas camadas dos detalhes de implementação real destes recursos. Este conceito é conhecido como encapsulamento de dados.

Quando a camada n de uma máquina se comunica com a camada n de outra máquina, coletivamente, as regras e convenções usadas nesse diálogo são conhecidas como protocolo da camada n .

Um protocolo define o formato e ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou no recebimento de uma mensagem ou outro evento (KUROSE, 2011).

Os protocolos de rede permitem tratar a comunicação através do computador, independente do hardware de rede utilizado, da mesma forma um protocolo permite que se

especifique ou compreenda uma comunicação de dados sem depender de conhecimentos minuciosos do hardware de um fornecedor específico.

A documentação para o trabalho da Internet, as propostas para protocolos novos ou revisados e os padrões de protocolos TCP/IP encontram-se em uma série de relatórios técnicos denominados RFCs (Request for Comments). Os RFCs podem ser curtos ou longos, podem abranger conceitos amplos ou detalhados e podem ser padrões ou simplesmente propostas para novos protocolos.

2.2 Camada de Interface de redes

O nível mais baixo do protocolo TCP/IP é a camada da interface de redes, ou camada de enlace. Esta camada é responsável pela aceitação de datagramas da camada de rede e por sua transmissão através de uma rede física específica.

Os serviços prestados por essa camada dependem do protocolo empregado. Alguns exemplos de protocolos da camada de enlace são Ethernet, Wi-Fi e PPP (Point-to-Point Protocol).

2.2.1 Ethernet

Comer (1998) considera a Ethernet como uma conexão no nível de enlace de dados entre máquinas. Os quadros Ethernet, que consistem nas informações transmitidas, possuem comprimentos variáveis sendo que nenhum menor que 64 octetos¹ ou superior a 1518 octetos.

Cada quadro começa com um Preâmbulo (**Preamble**) de 8 bytes, cada byte tem um padrão de bits 10101010, com exceção do último que acaba com os bits 11. Esta sequência de bits produz uma onda quadrada, com a intenção de permitir o sincronização do *clock* do receptor e do *clock* do transmissor. O último byte do preâmbulo é chamado de IDQ ou Início de Quadro (**Start Frame Delimiter**), e serve para sinalizar ao receptor que o restante do quadro está para começar.

Em seguida o quadro contém dois endereços MAC (Media Access Protocol), um para destino e um para a origem. Cada um deles possui 6 bytes de extensão. O primeiro bit transmitido do endereço de destino é 0 para endereços comuns e 1 para endereços de grupo.

¹Quantidade de oito bits presente em todos os computadores. (COMER, 1998)

Quando um quadro é enviado para um endereço de grupo, todas as estações do grupo o recebem, esta transmissão é chamada *multicasting*. O endereço que consiste em todos os bits 1 é reservado para o *broadcasting*, e é aceito por todas as estações da rede. Uma importante característica dos endereços MAC é que eles são globalmente exclusivos, atribuídos de forma centralizada pelo IEEE (Institute of Electrical and Electronics Engineers) para garantir que duas estações, em qualquer lugar do mundo, não tenham o mesmo endereço. Para isso os três primeiros bytes do endereço são usados para um identificador exclusivo da organização ou OUI (Organizationally Unique Identifier), estes bytes são definidos pelo IEEE e indicam o fabricante. Cabe ao fabricante atribuir e programar os três últimos bytes do endereço na interface de rede.

Após vem o quadro Tipo (**Type**) contendo um número inteiro de 16 bits que identifica o tipo de informação que está sendo transportado no quadro. Quando um quadro chega a um determinado equipamento, o tipo do quadro é utilizado para determinar qual é o protocolo que deve processar as informações contidas no campo Dados.

O campo Dados (**Data**) deve conter de 46 a 1500 bytes, e guarda as informações do datagrama da camada de rede. Sempre que as informações excederem 1500 bytes o *host*² de origem deve fragmentar o datagrama. O tamanho mínimo é de 46 bytes, um dos mais importantes motivos para a existência de um tamanho mínimo é impedir que uma estação conclua a transmissão de um quadro curto antes de o primeiro bit ter atingido a outra extremidade do cabo, o que poderia ocasionar uma colisão. Sempre que um quadro for menor que 46 bytes o campo Preenchimento (**Payload**) será usado para complementar o resto do quadro até que o mesmo tenha o tamanho mínimo.

O último campo Ethernet é o Checksum, um CRC (Cyclic Redundancy Check) de 32 bits. Ele é um código de detecção de erros utilizado para determinar se os bits do quadro foram recebidos corretamente. Caso um erro seja detectado, o quadro é descartado.

O formato do quadro Ethernet pode ser visualizado na Figura 2.

Figura 2 – Quadro Ethernet

Preâmbulo	I d q	Endereço de destino	Endereço de origem	Tipo	Dados	Preenchimento	Checksum
-----------	-------------	------------------------	-----------------------	------	-------	---------------	----------

Fonte: Adaptado pelo autor com base em Comer (1998).

²Qualquer sistema de computador de usuário final que se conecta a uma rede. (COMER, 1998)

2.3 Camada de Rede

A camada inter-rede ou *internet*, também chamada de camada de rede, integra toda a arquitetura, mantendo-a unida. Sua tarefa é permitir que os *hosts* transmitam pacotes em qualquer rede e garantir que o tráfego chegue até o destino. Ela também lida com datagramas³ de entrada, verificando sua validade, e utilizando algoritmos de roteamento para decidir se o datagrama deve ser processado no local ou deve ser enviado para outra máquina.

A camada de rede define um formato oficial para seus pacotes, o protocolo IP (Internet Protocol), mais um protocolo que o acompanha, chamado ICMP (Internet Control Message Protocol).

2.3.1 IP

O IP, Internet Protocol, é o protocolo da camada de rede que mantém a Internet unida. Ao contrário da maioria dos protocolos da camada de rede mais antigos, o IP foi projetado desde o início com o objetivo de interligar redes.

Este protocolo oferece três importantes definições. Primeiramente, define a unidade básica de transferência de dados (datagrama) utilizada em uma interligação em redes TCP/IP.

O protocolo IP também desempenha a função de roteamento, escolhendo um caminho por onde os dados serão enviados. E além disso o IP inclui um conjunto de regras que concentram a ideia de entrega não-confiável e sem conexão de pacotes.

Um serviço não-confiável não garante a entrega dos dados. O pacote pode ser perdido, reproduzido, atrasar-se ou ser entregue com problemas, porém o serviço não detectará tais condições, nem informará isso ao transmissor nem ao receptor.

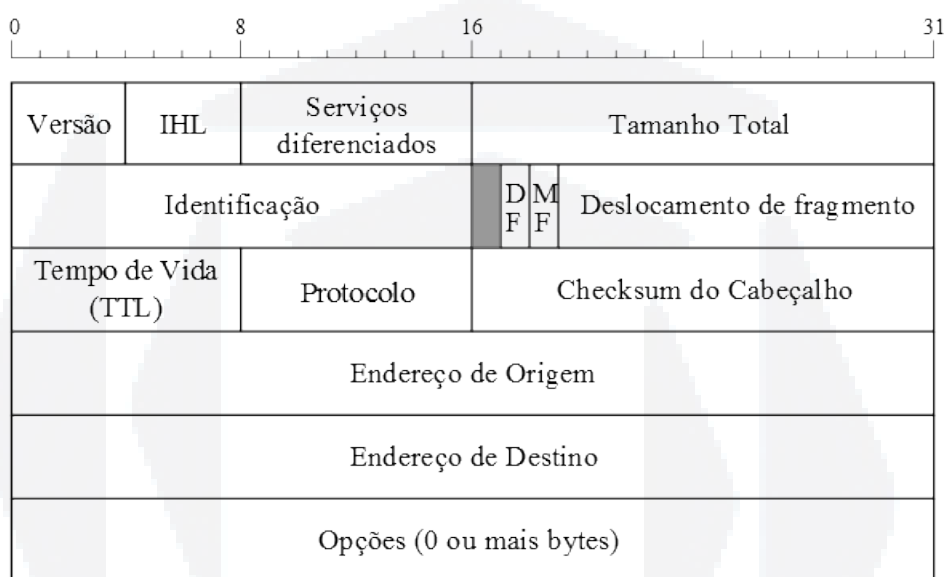
O IP é denominado sem conexão pois cada pacote é independente dos demais. Uma sequência de pacotes enviados de um computador a outro pode trafegar por caminhos diferentes. Alguns pacotes podem ser perdidos enquanto outros são entregues.

Numa rede de computadores, a camada de transporte recebe fluxos de dados e os divide para que possam ser enviados como pacotes IP. Na teoria estes pacotes podem ter até 64 KB, entretanto eles geralmente têm no máximo 1500 bytes.

³Unidade de informação básica passada através de uma interligação TCP. (COMER, 1998)

A versão do protocolo IP mais utilizada atualmente é a versão quatro e por esse motivo é denominada IPv4, esta versão é especificada segundo o RFC 791. O datagrama desta versão consiste em uma parte de cabeçalho e uma parte de dados. O cabeçalho tem uma parte fixa de 20 bytes e uma parte opcional de tamanho variável. A estrutura do cabeçalho do datagrama IP pode ser visualizada conforme a Figura 3.

Figura 3 – Cabeçalho do datagrama IP



Fonte: Adaptado pelo autor com base em Tanenbaum (2011).

Os principais campos do cabeçalho IP são descritos abaixo:

- Versão (**Version**): controla a versão do protocolo a qual o datagrama pertence;
- IHL (**Internet Header Length**): informa o tamanho do cabeçalho IP;
- Serviços diferenciados (**Differentiated Services**) ou Tipo de Serviço (**Type of service**): especifica como o datagrama deve ser tratado;
- Tamanho total (**Total length**): inclui tudo o que há no datagrama, cabeçalho e dados. Com um tamanho máximo de 65.535 bytes;
- Identificação (**Identification**): utilizado para determinar a qual datagrama pertence um fragmento recém-chegado;
- DF (**Don't Fragment**): ordem para não fragmentar o datagrama;
- MF (**More Fragments**): indica que há mais fragmentos deste datagrama;
- Deslocamento de fragmento (**Fragment Offset**): informa a que ponto do datagrama atual o fragmento pertence;

- TTL (**Time to Live**): contador utilizado para limitar a vida útil dos pacotes;
- Protocolo (**Protocol**): informa qual o protocolo da camada de transporte deve ser entregue o pacote;
- Checksum do Cabeçalho (**Header Checksum**): utilizado para detectar erros;
- Endereço de Origem (**Source address**) e Endereço de Destino (**Destination address**): indicam o endereço IP das interfaces de rede de origem e de destino, respectivamente;
- Opções (**Options**): projetado para que versões posteriores do protocolo incluam informações inexistentes no projeto original.

Um dos recursos que define o IPv4 são seus endereços de 32 bits, sendo que cada interface de rede possui um endereço IP. A maioria dos *hosts* de uma rede tem apenas uma interface e portanto apenas um endereço IP, porém alguns dispositivos possuem várias interfaces, como os roteadores, possuindo assim vários endereços IP.

Cada interface em cada hospedeiro e roteador da Internet global tem de ter um endereço IP globalmente exclusivo. Contudo, estes endereços não podem ser escolhidos de qualquer maneira.

Os endereços IPs são compostos de uma parte de rede, de tamanho variável, nos bits superiores e uma parte de *host* nos bits inferiores. A parte de rede tem o mesmo valor para todos os *hosts* em uma única rede, chamada de prefixo.

Estes endereços são escritos em notação decimal com ponto, onde cada um dos 4 bytes é escrito com números de 0 a 255, como por exemplo 128.208.2.151. O tamanho do prefixo é determinado pelo número de bits na parte de rede, os bits restantes fazem parte do campo de *host* e podem variar.

Como o tamanho do prefixo não pode ser deduzido apenas pelo endereço IP, os protocolos de roteamento devem transportar os prefixos aos roteadores. O tamanho do prefixo corresponde a uma máscara binária de 1s na parte destinada à rede, sendo chamada de máscara de sub-rede.

2.3.2 ICMP

O ICMP – Internet Control Message Protocol – foi acrescentado aos protocolos TCP/IP, inicialmente, para permitir que roteadores de uma interligação de redes informem os

erros ou forneçam informações sobre ocorrências inesperadas. Porém ele não se restringe aos roteadores, uma máquina arbitrária pode enviar uma mensagem ICMP a qualquer outra máquina.

O ICMP também é usado para testar a Internet. Cerca de doze tipos de mensagens ICMP são definidas. Cada tipo de mensagem é encapsulada dentro de um pacote IP, porém o ICMP não é considerado um protocolo de prioridade, ele é uma parte necessária do IP. O motivo de usar IP para conduzir mensagens ICMP é que elas precisam trafegar por várias redes físicas para alcançar seu destino final. Portanto, não podem ser entregues somente através do transporte físico.

Cada mensagem ICMP tem um formato próprio e o mesmo é definido segundo o RFC 792, entretanto todas começam com os mesmos três campos. O primeiro é o campo tipo de mensagem (**Type**), com oito bits inteiros, identifica a mensagem. O campo tipo de ICMP define o significado da mensagem, assim como o seu formato. Os tipos de mensagens, e seus respectivos códigos podem ser vistos na Tabela 1.

Tabela 1 – Tipos de mensagens do protocolo ICMP

Campo Tipo	Tipo de mensagem ICMP
0	Resposta ao eco (<i>Echo Reply</i>)
3	Destino não-acessível
4	Dissipação da origem
5	Redirecionamento (mudar rota)
8	Solicitação de eco (<i>Echo Request</i>)
11	Tempo excedido para um datagrama
12	Problema de parâmetro num datagrama
13	Solicitação de indicação de hora
14	Resposta de indicação de hora
15	Solicitação de informação (obsoleto)
16	Resposta de informação (obsoleto)
17	Solicitação de máscara de endereço
18	Resposta de máscara de endereço

Fonte: Comer (1998).

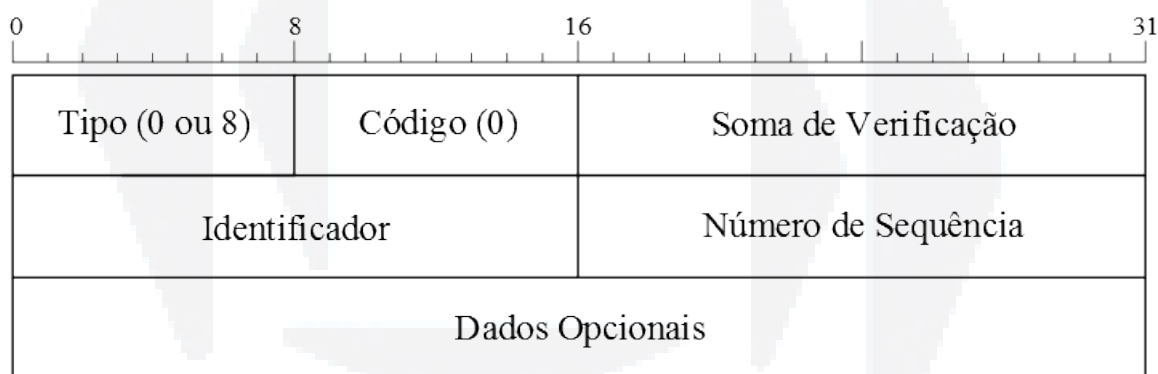
O segundo é o campo código (**Code**), com oito bits, que fornece informações adicionais sobre o tipo de mensagem. E o terceiro é um campo soma de verificação, de 16

bits. Além destes campos todas as mensagens ICMP sempre incluem o cabeçalho e os primeiros 64 bits de dados do datagrama que causou o problema.

Os protocolos TCP/IP fornecem recursos para auxiliar na identificação de problemas de rede. Uma das ferramentas mais utilizadas na depuração são as mensagens de solicitação de *echo* e de resposta de *echo*, em muitos sistemas operacionais o comando responsável pelo envio destas mensagens ICMP recebe o nome de *ping*.

Qualquer máquina que receba um *echo request* envia um *echo reply* ao transmissor. A solicitação contém uma área opcional de dados. A resposta contém uma cópia dos dados enviados na solicitação. A solicitação de *echo* e a resposta a ele associada podem ser usadas para testar se um receptor é acessível e se este está respondendo. O formato das mensagens de envio e solicitação de *echo* é demonstrado na Figura 4.

Figura 4 – Mensagem Echo (ICMP)



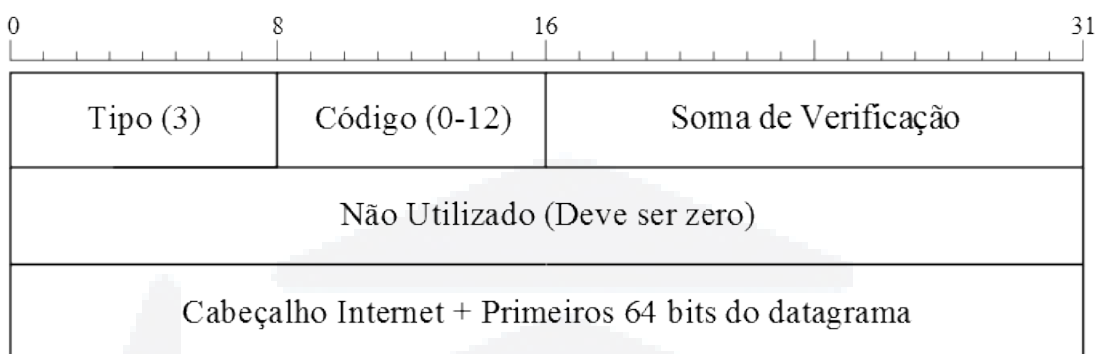
Fonte: Adaptado pelo autor com base em Comer (1998).

O campo Dados Opcionais (**Optional Data**) é um campo de tamanho variável contendo os dados enviados. Os campos Identificador (**Identifier**) e o Número de Sequência (**Sequence Number**) são usados pelo transmissor para combinar as solicitações com as respostas. O valor do campo Tipo define se a mensagem é um *echo request*, contendo o valor oito, ou um *echo reply*, com o valor zero.

Quando um roteador não consegue enviar ou entregar um datagrama IP, ele envia uma mensagem ICMP de destinatário inacessível, a estrutura da mensagem é apresentada na Figura 5. Embora o IP seja um mecanismo de transmissão sem garantia, o descarte de pacotes não deve ser encarado com descaso. Sempre que um erro impedir que um roteador roteie ou

entregue um datagrama, o mesmo transmite uma mensagem de destinatário indisponível e após descarta o datagrama.

Figura 5 – Mensagem Destinatário Não Acessível (ICMP)



Fonte: Adaptado pelo autor com base em Comer (1998).

O campo código de uma mensagem de destinatário inacessível contém um número inteiro que descreve o problema. Os possíveis valores deste campo, juntamente com o seu significado pode ser visualizado na Tabela 2.

Tabela 2 – Possíveis valores de códigos para uma mensagem de destinatário não acessível do protocolo ICMP

Valor do Código	Significado
0	Rede inacessível
1	<i>Host</i> inacessível
2	Protocolo inacessível
3	Porta inacessível
4	Fragmentação necessária e DF
5	Rota de origem em falha
6	Rede de destino desconhecida
7	<i>Host</i> de destino desconhecido
8	<i>Host</i> de origem isolado
9	Comunicação com rede destinatária administrativamente proibida
10	Comunicação com <i>host</i> destinatário administrativamente proibida
11	Rede inacessível para tipo de serviço
12	<i>Host</i> inacessível para tipo de serviço

Fonte: Comer (1998).

2.4 Camada de Transporte

A camada de transporte é responsável por prover a comunicação de um programa aplicativo para outro, sendo que tal comunicação é chamada fim-a-fim. Esta camada pode regular o fluxo de informações, fornecer transporte confiável – assegurando assim que os dados cheguem sem erros e em sequência. O software da camada de transporte divide o fluxo de dados transmitidos em pequenas partes (usualmente chamadas de pacotes) e passa cada pacote, juntamente com o endereço de destino, à camada seguinte (a camada de rede) para ser transmitido.

Os dois principais protocolos da camada de transporte são o TCP (Transmission Control Protocol) e o UDP (User Data Protocol). O TCP é orientado a conexões e confiável, permitindo a entrega sem erros de um fluxo de bytes. Já o protocolo UDP não estabelece conexões, não sendo portanto confiável, mais ainda assim útil para aplicações que não desejam a entrega em sequência ou o controle de fluxo de dados.

2.4.1 UDP

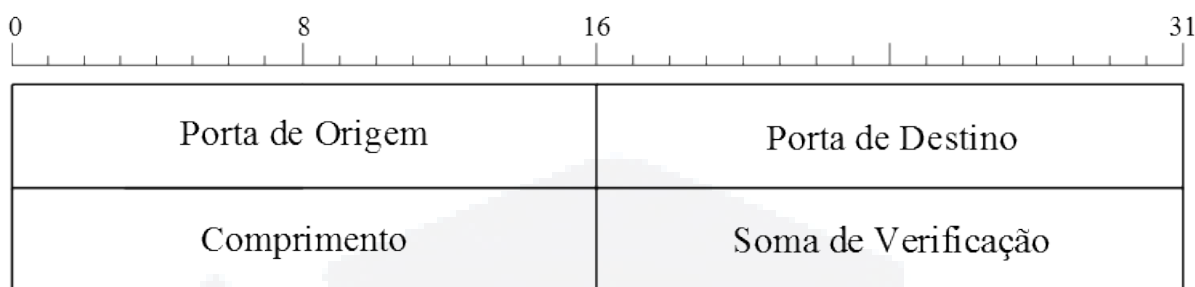
Na pilha de protocolos TCP/IP, o User Datagram Protocol, ou UDP, fornece o principal mecanismo utilizado pelos programas aplicativos para o envio de datagramas a outros programas iguais. Além dos dados enviados, cada mensagem UDP contém um número de porta de destino e também um número de porta de origem. Sem as informações das portas, a camada de transporte não saberia o que fazer com o pacote que chega, com estas informações a camada entrega os dados à aplicação correta.

Semelhante ao IP, o UDP também fornece uma conotação não confiável de transmissão de datagramas sem conexão, cabendo este controle a aplicação que o utiliza. O UDP não faz uso de confirmações para certificar-se que as mensagens foram corretamente enviadas, não ordena as mensagens e nem oferece o controle da velocidade que as informações são transmitidas. Desta forma, os pacotes podem se perder, serem duplicados, chegar com problemas ou chegar mais rápido do que podem ser processados pelo destinatário.

Cada mensagem UDP é conhecida com um datagrama de usuário. Em tese, um datagrama de usuário consiste em duas partes: um cabeçalho UDP e uma área de dados UDP.

Como mostrado na Figura 6, o cabeçalho está dividido em quatro campos de 16 bits, descritos abaixo e definidos pelo RFC 768.

Figura 6 – Cabeçalho UDP



Fonte: Adaptado pelo autor com base em Comer (1998).

Os campos Porta de Origem (**Source Port**) e Porta de Destino (**Destination Port**) contêm os números de porta do protocolo UDP de 16 bits usados para demultiplexar os datagramas entre os processos que esperam para recebê-los. A porta de origem é opcional, e serve para especificar para qual porta deve ser enviada as respostas.

O campo Comprimento (**Length**) contém uma contagem de octetos do datagrama UDP, incluindo o cabeçalho UDP e os dados do usuário, desta forma o valor mínimo deste campo é de 8 bytes, que representa o tamanho do cabeçalho e o valor máximo é de 65515 bytes, devido ao limite de tamanho do protocolo IP.

A Soma de Verificação (**Checksum**) UDP é opcional e não precisa ser usada, um valor zero no campo Soma de Verificação significa que a soma não foi calculada.

2.4.2 TCP

O protocolo de controle de transmissão, ou TCP, foi projetado para oferecer um fluxo de bytes fim a fim confiável em uma rede integrada não confiável. Uma rede integrada pode ter topologias, larguras de banda, atrasos, tamanhos de pacotes e outros parâmetros completamente diferentes em cada uma de suas partes. O TCP se adapta dinamicamente às propriedades de uma rede integrada e é robusto diante dos muitos tipos de falhas que podem ocorrer.

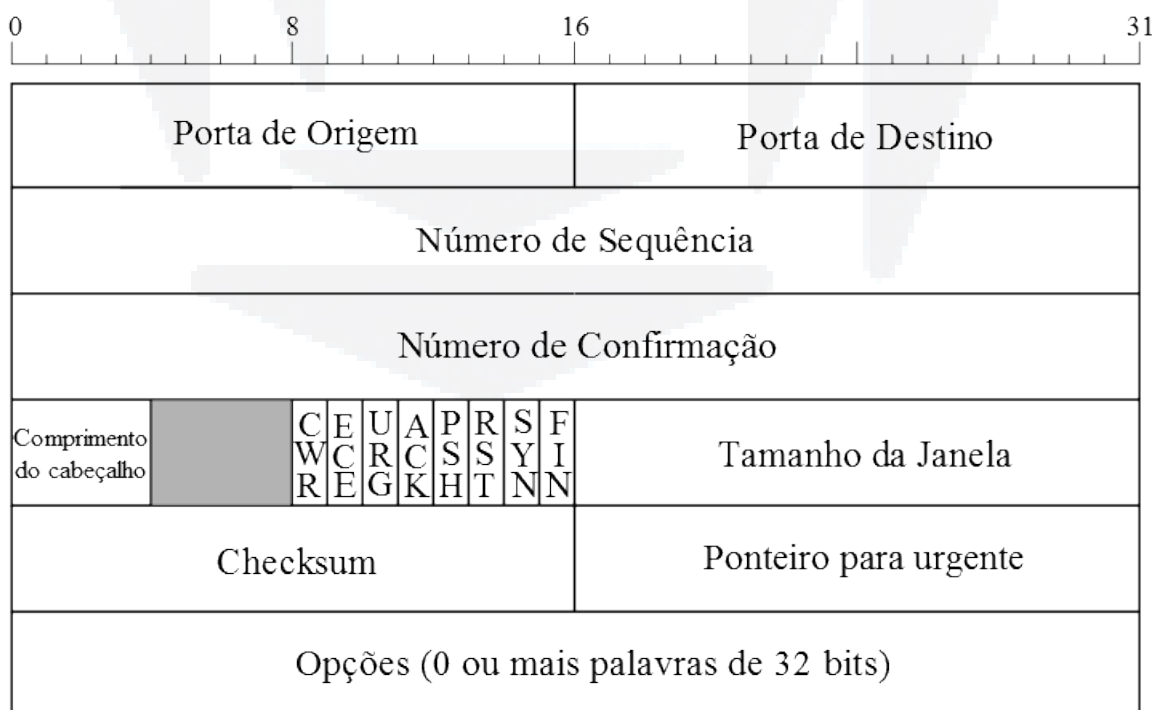
A camada de rede não oferece nenhuma garantia que os datagramas serão entregues da forma apropriada, nem indicação alguma de velocidade em que os datagramas devem ser enviados. Cabe ao TCP enviar datagramas e controlar a velocidade para um melhor

desempenho da rede, sem que isso cause congestionamento. Além disso os datagramas podem vir fora de ordem e é papel do TCP reorganizá-los.

O serviço TCP é obtido quando tanto o transmissor quanto o receptor criam pontos externos chamados *sockets*. Cada *socket* tem um número que consiste no endereço IP do *host* e um número de 16 bits local para este *host*, chamado porta. Para o serviço TCP funcionar é necessário que seja estabelecida uma conexão entre um *socket* da máquina transmissora e um *socket* da máquina receptora.

As entidades transmissora e receptora do TCP trocam dados na forma de segmentos, o formato dos segmentos é definido no RFC 793. Um segmento TCP consiste em um cabeçalho fixo de 20 bytes mais uma quantidade variável de bytes para as opções, seguidos por zero ou mais bytes de dados, como pode ser observado na Figura 7. O TCP define qual deve ser o tamanho dos seguimentos, sendo que o tamanho máximo deve caber em na carga útil do IP. Porém na prática o que define o limite superior de tamanho dos seguimentos é a MTU (Maximum Transfer Unit) da rede, que geralmente tem 1500 bytes.

Figura 7 – Cabeçalho TCP



Fonte: Adaptado pelo autor com base em Tanenbaum (2011).

Os campos Porta de Origem (**Source Port**) e Porta de destino (**Destination Port**) identificam os pontos terminais de uma conexão. O campo Número de Sequência (**Sequence Number**) identifica a posição no stream de bytes do transmissor dos dados do segmento e portanto ordena a sequência dos segmentos. E o campo Número de Confirmação (**Acknowledgment Number**) identifica o número do octeto que a origem espera receber depois do envio do segmento.

O campo Comprimento do cabeçalho (**Data Offset**) TCP informa quantas palavras de 32 bits existem no cabeçalho. Esta informação é importante pois o campo Opções (**Options**) tem um tamanho variável, fazendo com que o cabeçalho tenha um tamanho variável.

Após o cabeçalho contém oito flags de 1 bit. As flags ECE e CWR são utilizadas no controle de congestionamento. O valor 1 no campo URG significa que os dados têm alta prioridade e necessitam ser processados imediatamente, porém este recurso é raramente usado. Caso esta flag seja usada o campo Ponteiro para urgente (**Urgent Pointer**) é usado para informar a posição, a partir do número de sequência atual, em que os dados urgentes se encontram.

A flag ACK com o valor 1 indica que o Número de Confirmação é válido. Caso o ACK seja igual a zero significa que o segmento não tem confirmação e o Número de Confirmação é ignorado.

A flag PSH indica dados com PUSH. Com esta opção o receptor entrega os dados à aplicação logo após sua chega, ao invés de guardá-los até que um buffer completo fosse preenchido.

A flag RST é utilizada para reiniciar a conexão que tenha ocorrido alguma falha. Outras utilizações desta flag são para rejeitar um segmento inválido ou para recusar uma tentativa de conexão. Normalmente quando o bit RST está ativo significa que ocorreu um problema.

A flag SYN é usada para estabelecer uma conexão. Enquanto a flag FIN é utilizada para encerrar uma conexão, indicando que o transmissor não tem mais dados para ser enviados. Tanto o segmento SYN quanto o segmento FIN têm números de sequência e por esta razão são processados na ordem correta.

O campo Tamanho da janela (**Window Size**) controla o tamanho da janela deslizante do TCP indicando quantos bytes podem ser enviados a partir do byte confirmado. Quando

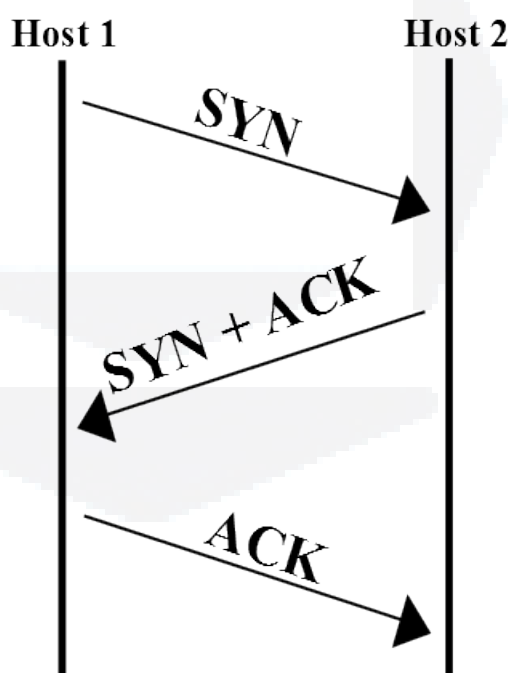
este campo tiver o valor 0 significa que todos os bytes até o momento foram recebidos, porem o receptor não pode mais receber nenhum dado no momento.

O **Checksum** é utilizado para aumentar a confiabilidade do protocolo. O checksum é calculado com base nas informações do cabeçalho, dos dados e de um pseudocabeçalho que depende da versão do protocolo IP em que o segmento TCP está encapsulado.

O campo Opções (**Options**) foi projetado para oferecer recursos extras, que não foram previstos no projeto original do TCP. Muitas opções foram definidas e várias são comumente usadas. As opções têm tamanhos variáveis, sendo sempre múltiplos de 32 bits, podendo se estender para até 40 *bytes*, quando o tamanho total do cabeçalho chega ao máximo suportado.

As conexões TCP são estabelecidas por meio do *handshake* de três vias, demonstrado na Figura 8. Este protocolo de estabelecimento de conexão envolve um *peer* verificando com o outro se a solicitação de conexão esta realmente ativa.

Figura 8 – *Handshake* de estabelecimento da conexão TCP



Fonte: Adaptado pelo autor com base em Comer (1998).

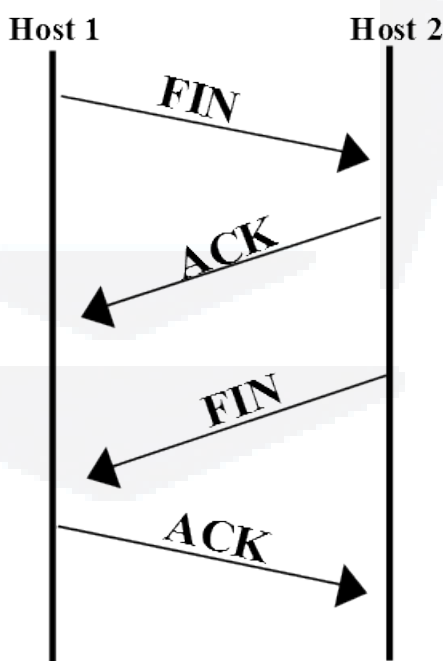
Para estabelecer a conexão um dos lados aguarda passivamente por um conexão de entrada executando as primitivas **LISTEN** e **ACCEPT**, nesta ordem. O outro lado executa a primitiva **CONNECT**, especificando o IP e a porta a qual deseja se conectar, além do tamanho

máximo do segmento TCP que deseja utilizar. A primitiva CONNECT envia um segmento TCP com o bit da *flag* SYN ativado e o bit da *flag* ACK desativado.

Se algum processo estiver na escuta da porta, o mesmo receberá o segmento TCP de entrada. Em seguida ele poderá decidir se aceitará ou rejeitará a conexão. Caso aceite a conexão o receptor envia um segmento com os bits SYN e ACK ativos. E por fim o transmissor responderá com um segmento ACK ativo, completando o *handshake* de três vias.

Para finalizar uma conexão, qualquer um dos lados pode enviar um segmento com o bit FIN ativo, informando que não há mais dados a serem enviados. Quando o FIN é recebido no outro lado, é enviado um segmento com o bit ACK ativo, confirmando que a conexão foi finalizada. Entretanto os dados podem continuar fluindo normalmente no sentido contrário, para encerrar de vez a conexão é necessário que se repita este processo em ambos os lados. A troca de segmentos para finalização da conexão pode ser vista mais claramente na Figura 9.

Figura 9 – Troca de mensagens para finalização da conexão TCP



Fonte: Adaptado pelo autor com base em Comer (1998).

As etapas necessárias para o estabelecimento e o encerramento de conexões podem ser melhor representadas por uma máquina de estados finita com 11 estados. Os estados são descritos na Tabela 3.

Tabela 3 – Estados da máquina de estados do TCP

Estado	Descrição
CLOSED	Nenhuma conexão ativa ou pendente
LISTEN	O servidor está esperando a chegada de uma chamada
SYN RCVD	Uma solicitação de conexão chegou; espera por ACK
SYN SENT	A aplicação começou a abrir uma conexão
ESTABLISHED	O estado normal para a transferência de dados
FIN WAIT 1	A aplicação informou que terminou de transferir
FIN WAIT 2	O outro lado concordou em encerrar
TIME WAIT	Aguarda a entrega de todos os pacotes
CLOSING	Ambos os lados tentaram encerrar a conexão simultaneamente
CLOSE WAIT	O outro lado deu início a um encerramento
LAST ACK	Aguarda e entrega de todos os pacotes

Fonte: Tanenbaum (2011).

Do ponto de vista do cliente, a conexão inicia no estado CLOSED. Quando um programa de aplicação na máquina emite uma solicitação CONNECT é criado um registro da conexão e o estado da conexão passa para SYN SENT, enviando um segmento SYN. Se for recebido o segmento SYN + ACK o TCP envia o ACK finalizando o *handshake* e passando para o estado de ESTABLISHED. Somente após isso os dados podem ser enviados e recebidos.

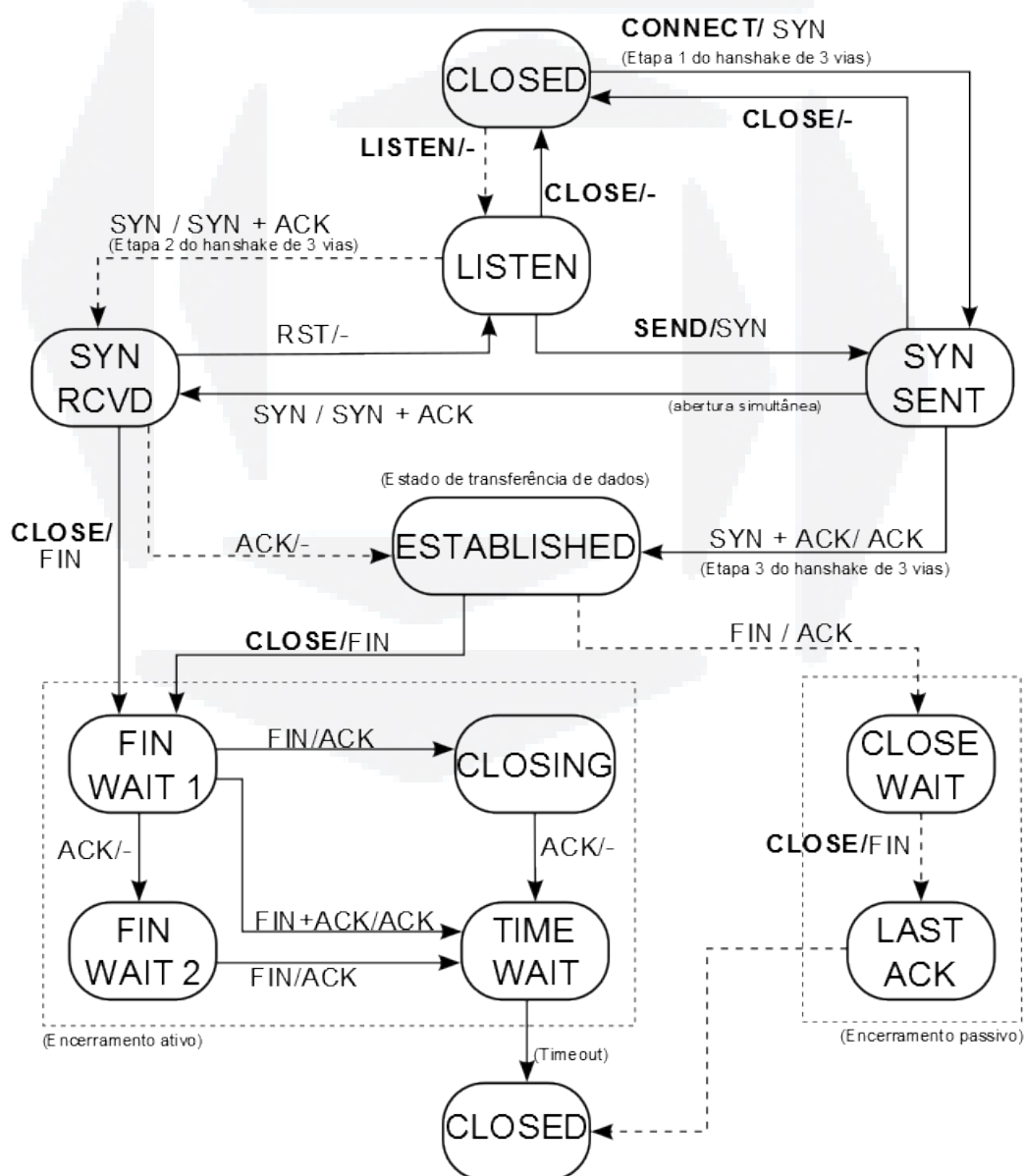
Caso seja desejado finalizar a conexão a primitiva CLOSE é executada, enviando um segmento FIN e aguardando o retorno de um ACK. Quando o ACK é recebido, há uma transição para o estado FIN WAIT 2 e um sentido da conexão é desativado. Quando o outro lado for desativado chegará um FIN e o cliente responderá com um ACK, neste momento o TCP aguarda um período para se certificar que todos os pacotes foram recebidos e então a conexão será finalizada, voltando ao estado CLOSED.

Do ponto de vista só servidor, a conexão também inicia no estado CLOSED. Após o início o servidor executa uma primitiva LISTEN e aguarda por um segmento SYN. Quando o SYN chegar a conexão será confirmada e o servidor passará para o estado SYN RCVD. Após a confirmação do SYN do servidor o *handshake* será finalizado e o estado passará para ESTABLISHED, quando a transferência de dados já pode ser feita normalmente.

Ao finalizar a transferência dos dados o cliente executará uma primitiva CLOSE, enviando um FIN para o servidor. Em seguida o servidor também executa a primitiva CLOSE, enviando um FIN para o cliente. Após o recebimento do ACK vindo do cliente, o servidor encerrará a conexão.

A máquina de estados do TCP pode ser vista na Figura 10. Na imagem as linhas contínuas mostram as ações exercidas por um cliente TCP, enquanto as linhas tracejadas representam as ações do servidor TCP.

Figura 10 – Máquina de estados do protocolo TCP



Fonte: Adaptado pelo autor com base em Tanenbaum (2011).

2.5 Camada de Aplicação

O nível mais alto do protocolo TCP/IP é a camada de aplicação, onde os usuários rodam programas aplicativos que acessam serviços disponíveis através de uma interligação da rede. Um aplicativo interage como um dos protocolos do nível de transporte para enviar e receber dados. Cada aplicativo escolhe o estilo de transporte necessário, que pode ser uma sequência de mensagens individuais ou um *stream* de bytes contínuos. Alguns dos mais importantes protocolos desta camada são o DNS (Domain Name Service), o HTTP (HyperText Transfer Protocol) e o RTP (Real-time Transport Protocol) utilizado para entrega de mídia em tempo real.

2.5.1 HTTP

O HTTP, HyperText Transfer Protocol, é um protocolo simples, do tipo solicitação-resposta, que roda sobre o TCP. Ele especifica quais mensagens os clientes podem enviar para os servidores e quais respostas recebem de volta. Segundo Kurose (2011), ele é implementado em dois programas: um programa cliente e outro servidor.

O modo habitual de um navegador entrar em contato com um servidor é estabelecer uma conexão TCP para a porta 80 da máquina servidora. Antigamente, com o HTTP 1.0 depois que a conexão era estabelecida, uma única solicitação era enviada e uma única resposta era devolvida.

Após o lançamento do HTTP 1.1, o protocolo passou a admitir conexões persistentes. Com elas, é possível estabelecer uma conexão TCP, enviar uma solicitação e obter uma resposta, e depois enviar solicitação adicionais e receber respostas adicionais. Esta estratégia também é chamada reuso de conexão.

As especificações do HTTP 1.1, seguindo o RFC 2616, definem os formatos das mensagens HTTP. Cada solicitação consiste em uma ou mais linhas de texto ASCII, sendo a primeira chamada de linha de requisição e as subsequentes linhas de cabeçalho. A linha de requisição tem três campos: o campo de método, o do URL e o da versão do HTTP. Os métodos internos são listados abaixo:

GET: Lê uma página *web*;

HEAD: Lê um cabeçalho de página *web*;

POST: Acrescenta algo a uma página *web*;
PUT: Armazena uma página *web*;
DELETE: Remove uma página *web*;
TRACE: Ecoa a solicitação recebida;
CONNECT: Conecta através de um *proxy*;
OPTIONS: Consulta opções para uma página.

Um exemplo de uma solicitação HTTP pode ser vista na Figura 11.

Figura 11 – Requisição HTTP

```
GET /somedir/page.html
HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/4.0
Accept-language: fr
```

Fonte: Adaptado pelo autor com base em Kurose (2011).

Toda solicitação obtém uma resposta que pode ser dividida em três seções. A primeira é a linha de estado e contém a versão do protocolo, um código de três dígitos, informando o estado da solicitação e uma mensagem de estado correspondente. Os códigos, com seus significados podem ser vistos na Tabela 4.

Tabela 4 – Possíveis códigos de uma resposta HTTP

Código	Significado
1xx	Informações (raramente são usadas)
2xx	Solicitação foi tratada com sucesso
3xx	Informa que o cliente deve procurar em outro lugar
4xx	A solicitação falhou devido a um erro do cliente
5xx	A solicitação falhou devido a um erro do servidor

Fonte: Tanenbaum (2011).

Após a linha de estado, pode vir uma ou mais linhas de cabeçalho, chamados de cabeçalhos de resposta. As linhas de cabeçalho dependem do tipo e versão do *browser*, bem como das configurações do usuário para o *browser*, como o idioma definido.

Por último vem o corpo da mensagem, contendo o conteúdo da mensagem solicitada pelo cliente, um exemplo de uma resposta HTTP pode ser vista na Figura 12.

Figura 12 – Resposta HTTP

```
HTTP/1.1 200 OK
Connection: close
Date: Sat. 07 Jul 2007 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Sun. 6 May 2007 09:23:24 GMT
Content-Length: 6821
Content-Type: text/html

(data data data data data ...)
```

Fonte: Adaptado pelo autor com base em Kurose (2011).

3 TRABALHOS RELACIONADOS

Neste capítulo são discutidos trabalhos relacionados com esta proposta, em particular um simulador de redes, profissional, disponibilizado pela empresa Cisco. Além disso serão citados outros simuladores que são muito complexos para alunos iniciantes, inclusive alguns deles voltados para a área de pesquisa acadêmica.

3.1 Simuladores comerciais e para pesquisa

O GLOMOSIM (2012) é um simulador de redes de alta fidelidade, que utiliza execução paralela para aumentar sua velocidade, sendo utilizado para o detalhamento de redes de grande porte. Este simulador é muito complexo, exigindo conhecimentos específicos de redes e de configuração. Este software é licenciado, porém existe uma versão acadêmica do mesmo.

O OPNET (2012) é um simulador com muitos recursos, utilizado em grandes empresas e operadoras de telecomunicações. Existe a versão acadêmica deste simulador, porém não é ideal para o ensino de alunos iniciantes já que seu uso não é trivial.

Segundo o fabricante, o NCTUns (2012) possui uma concepção moderna e inovadora, o que o torna interessante não só pelo seu alto desempenho mas também pelo seu aspecto pedagógico ou didático. Simula um conjunto amplo de protocolos, tanto para redes móveis como para redes fixas. Assim, a interface gráfica com o usuário é a sua grande vantagem. Como desvantagens é possível citar as dificuldades de implantação, bem como sua portabilidade para outros sistemas operacionais.

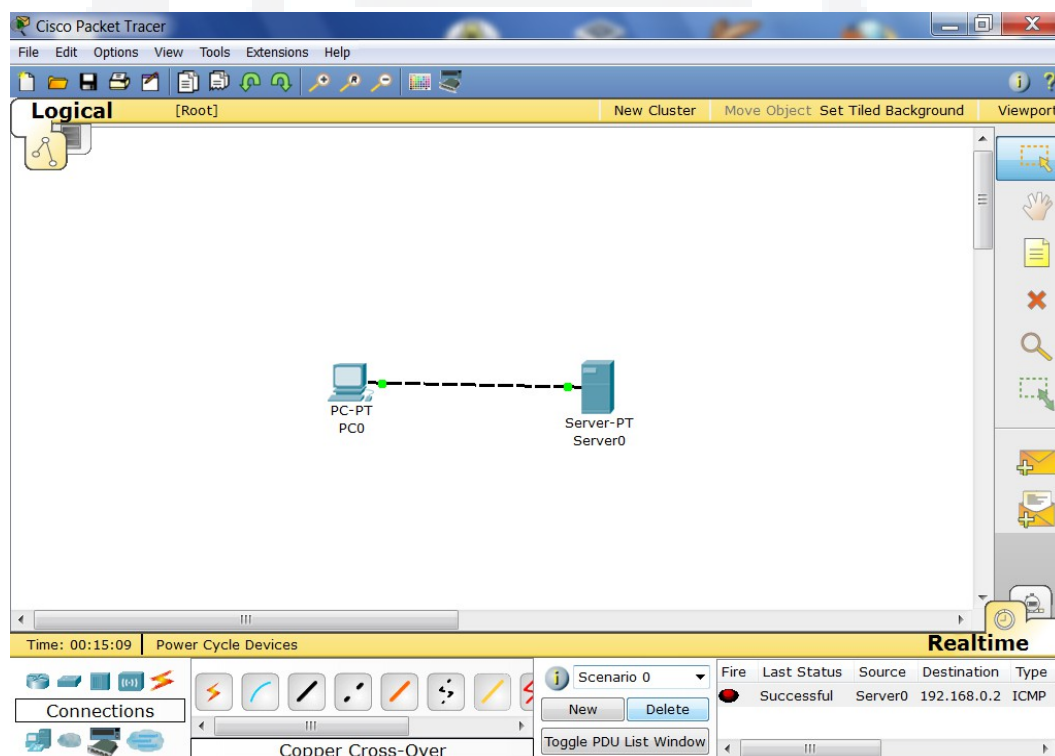
O NS (2012) é utilizado principalmente por pesquisadores, por ter distribuição gratuita e código aberto. Tal fato o torna adequado a situações onde é necessário desenvolver novas

funcionalidades, como em teses e projetos de pesquisa aplicada. No entanto, a sua interface não é amigável ao usuário. Além disso, os protocolos e tecnologias no NS em geral são desenvolvidos para uso isolado, para resolução de problemas específicos.

3.2 Simulador Cisco Packet Tracer

O Packet Tracer é um programa gratuito fornecido pela Cisco que permite simular uma rede de computadores. Possibilitando a criação e visualização da transmissão de pacotes virtuais através da rede virtual criada em tempo real. Na Figura 13 é possível verificar a interface do simulador de redes disponibilizado pela Cisco.

Figura 13 – Cisco Packet Tracer



Fonte: Elaborado pelo autor.

Na parte inferior do simulador se encontram os dispositivos disponíveis para criar a rede virtual. Além de *hosts* comuns, é possível incluir *switch*, *hubs*, roteadores, servidores entre outros.

Ainda na parte inferior encontram-se dois importantes recursos. O primeiro é o acompanhamento das mensagens criadas, contendo informações como máquina de destino e máquina de origem, protocolo utilizado, status do envio e mais algumas outras opções. O

segundo é uma área para simular o envio dos pacotes, nela é possível ver todo o caminho que os pacotes fazem, além de poder ver os detalhes de cada pacote e uma explicação detalhada sobre cada um deles.

A direita do simulador têm as opções de criação de pacote. Existem duas formas básicas para criar um pacote. É possível criar um pacote simples, que consiste em uma mensagem ICMP, onde é necessário selecionar primeiramente a máquina de origem e após a máquina de destino e então o pacote é criado.

O simulador também disponibiliza a criação de pacotes de outros tipos de protocolos, para isso existe a opção de criar pacotes complexos. Nesta opção é preciso clicar no remetente, após isso é aberta uma nova janela, demonstrada na Figura 14, com as opções do pacote. As opções dependem da camada em que se encontra o protocolo, porém as principais informações necessárias são o IP de origem e o IP de destino.

Figura 14 – Janela para criação de uma mensagem complexa

Create Complex PDU

Source Settings

Source Device: PC0

Outgoing Port: FastEthernet ☒ Auto Select Port

PDU Settings

Select Application: HTTP

Destination IP Address: 192.168.0.6

Source IP Address: 192.168.0.2

TTL: 32

TOS: 0

Source Port: 9000

Destination Port: 80

Size: 0

Simulation Settings

☒ One Shot Time: 400 Seconds

☐ Periodic Interval: Seconds

Create PDU

Fonte: Elaborado pelo autor.

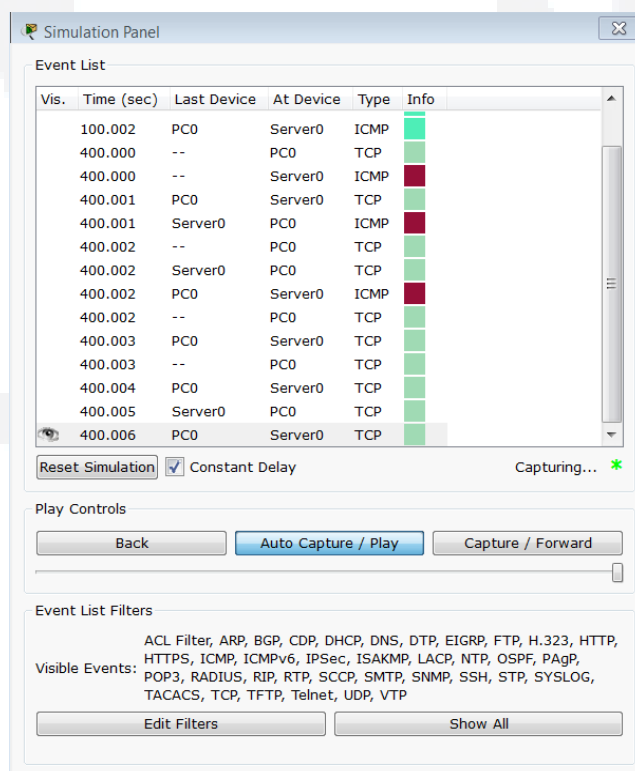
Para testar a ferramenta foi criada uma topologia simples de rede, onde uma máquina exerce o papel de cliente e outra o papel de um servidor genérico. Ambas são interligadas através de um cabo *cross*, dispensando assim a utilização de um *switch* ou *hub* para a troca de mensagens.

Para funcionar a comunicação entre os *hosts* foi necessário definir os IPs dos componentes. No exemplo a máquina cliente ficou com o IP 192.168.0.2 enquanto o servidor ficou com o IP 192.168.0.6. Ambas as máquinas ficaram com a máscara de rede 255.255.255.0.

Com os pontos da rede configurados, foi criada uma mensagem HTTP que seria enviada do cliente para o servidor. Para a criação da mensagem é preciso clicar no botão “Add Complex PDU” e então é aberta uma nova janela onde é possível definir os parâmetros da mensagens.

Na área de simulação é possível visualizar todas as mensagens trocadas entre as máquinas da rede. Tendo a opção de controlar o fluxo pacote por pacote ou então executar a transferência das mensagens de forma automática, onde o simulador envia um pacote a cada período de tempo. A lista das mensagens trocadas pode ser visualizada na Figura 15.

Figura 15 – Lista das mensagens trocadas

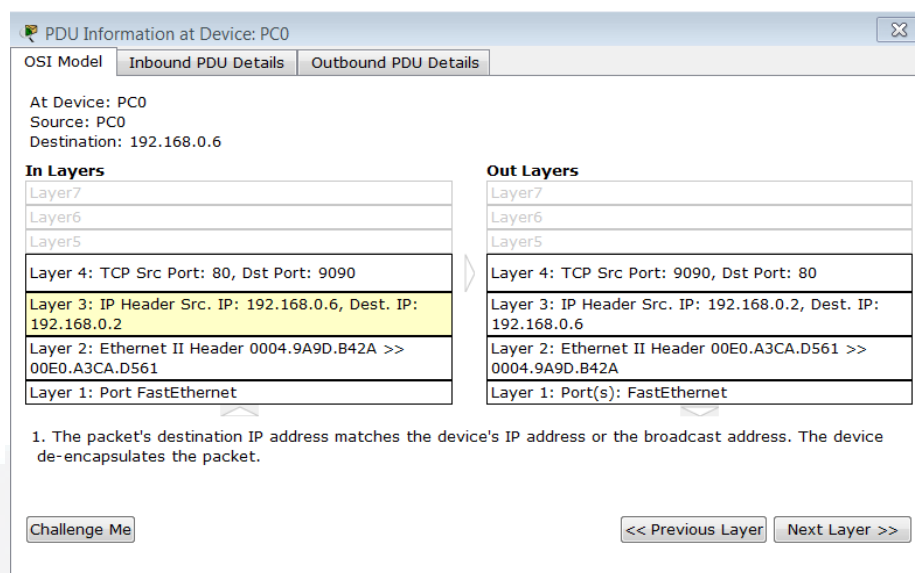


Fonte: Elaborado pelo autor.

Ao clicar em cima de uma das mensagens é possível visualizar os detalhes do pacote, sendo divididos em três abas. Na primeira aba é mostrada cada um dos protocolos utilizados, separados em cada uma das camadas do modelo OSI. Clicando em cima da camada é

mostrada uma explicação detalhada do papel que a camada esta exercendo. A aba com os detalhes das camadas do modelo OSI pode ser vista na Figura 16.

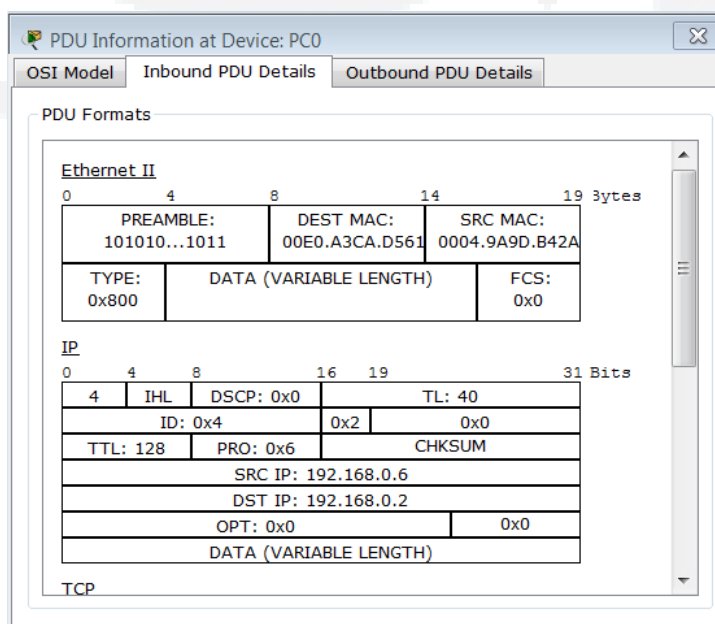
Figura 16 – Detalhes das camadas do modelo OSI



Fonte: Elaborado pelo autor.

As duas outras abas mostram os detalhes dos cabeçalhos de cada um dos protocolos utilizados para o transporte da mensagem. A segunda aba mostra os detalhes da mensagem de entrada do dispositivo, caso ele não exista esta aba não é mostrada.

Figura 17 – Detalhamento de um pacote enviado



Fonte: Elaborado pelo autor.

A terceira aba mostra os cabeçalhos da mensagem de resposta, ou seja, a mensagem enviada pelo dispositivo. A forma de visualização deste conteúdo é semelhante à demonstrada na Figura 17.

O simulador disponibilizado pela Cisco mostra com detalhes as estruturas dos principais protocolos do modelo TCP/IP. Porém, como foi descrito anteriormente, é necessário configurar uma rede virtual para poder simular o envio dos pacotes, o que torna seu uso complexo, dificultando o uso deste simulador por alunos iniciantes.

Uma vantagem que o simulador desenvolvido pelo autor tem em relação ao da Cisco é o fato dela ser uma ferramenta distribuída, permitindo a interação entre vários simuladores distintos, tornando-a mais didática e interativa. Ainda é possível modificar a maioria dos valores dos campos dos protocolos, verificando o impacto que estas alterações causa na comunicação ao contrário do simulador disponibilizado pela Cisco onde o conteúdo dos pacotes é estático sendo possível alterar poucas informações como os endereços de origem e destino e as portas de conexão.

Além disso o Cisco Packet Tracer é uma ferramenta fechada, não sendo possível modificar ou acrescentar novos recursos. O simulador proposto neste trabalho tem o código fonte aberto podendo ser alterado de acordo com as necessidades do professor.

4 IMPLEMENTAÇÃO

Neste capítulo é dada uma visão geral das melhorias desenvolvidas neste trabalho, assim como uma descrição da estrutura das classes, inicialmente criada por Müller (2010). Adicionalmente, são detalhadas as modificações que aprimoraram o simulador.

4.1 Visão geral

Para dar continuidade ao trabalho de Müller (2010), foram definidos alguns requisitos a serem desenvolvidos, além de terem sido feitas pequenas alterações de funções já desenvolvidas.

Um dos principais requisitos é a criação de um mecanismo de controle dos estados de um protocolo. Como entre os protocolos implementados somente o TCP possui estados, esta funcionalidade foi desenvolvida somente para este protocolo. Para o controle da transição entre os estados foi levado em consideração somente os valores dos bits de controle (ACK, SYN, FIN), sendo ignorados os valores dos demais campos.

Outra funcionalidade implementada foi a resposta automática para alguns protocolos, como o ICMP, TCP e HTTP. Também foi feita a resposta dos protocolos IP e Ethernet, pois os demais protocolos são encapsulados neles.

Para permitir que o simulador seja utilizado sem que o usuário esteja conectado a uma rede ou rodando uma máquina virtual, foi permitida a execução de múltiplas instâncias do simulador em uma mesma máquina pois anteriormente era possível executar somente uma. O número de estações ficou limitada em duas, para não sobrecarregar a rede e não ficar muito complexo para a gerência do usuário.

Com o intuito de tornar o simulador mais didático foram incluídos textos descritivos para cada um dos protocolos e de seus campos, explicando suas principais funcionalidades. Além disso foi criada uma tela com a tabela ARP (Address Resolution Protocol), para poder abordar conceitos como endereços físicos e de rede.

Ainda foi implementado um novo protocolo da camada de redes, o ICMP. Visto que este é um importante protocolo do modelo TCP/IP, sendo utilizado, dentre outras coisas, para verificar o estado da rede, como visto na Seção 2.3.2.

Pelo fato de grande parte das nomenclaturas e documentações da área de redes ser em inglês, foi criada uma função de internacionalização, permitindo que o usuário escolha em qual idioma ele prefere visualizar a aplicação.

Por último ainda foi alterada a forma de escolha dos protocolos que compõem um pacote. Anteriormente era necessário escolher um protocolo para cada camada do modelo TCP/IP, porém em alguns casos um pacote não contém protocolos da camada de rede ou aplicação como por exemplo o ICMP.

4.2 Estrutura das Classes

A estrutura do simulador é basicamente dividida em três tipos de classes: classes responsáveis pela interação e funcionamento do programa, classes que representam datagramas e classes que representam painéis de detalhes dos protocolos.

Estas classes foram divididas em dois pacotes, o pacote **simulador** e o pacote **datagrams**. No pacote **simulador** estão a maioria das classes do sistema e no pacote **datagrams** estão as especializações das classes abstratas **Datagram**.

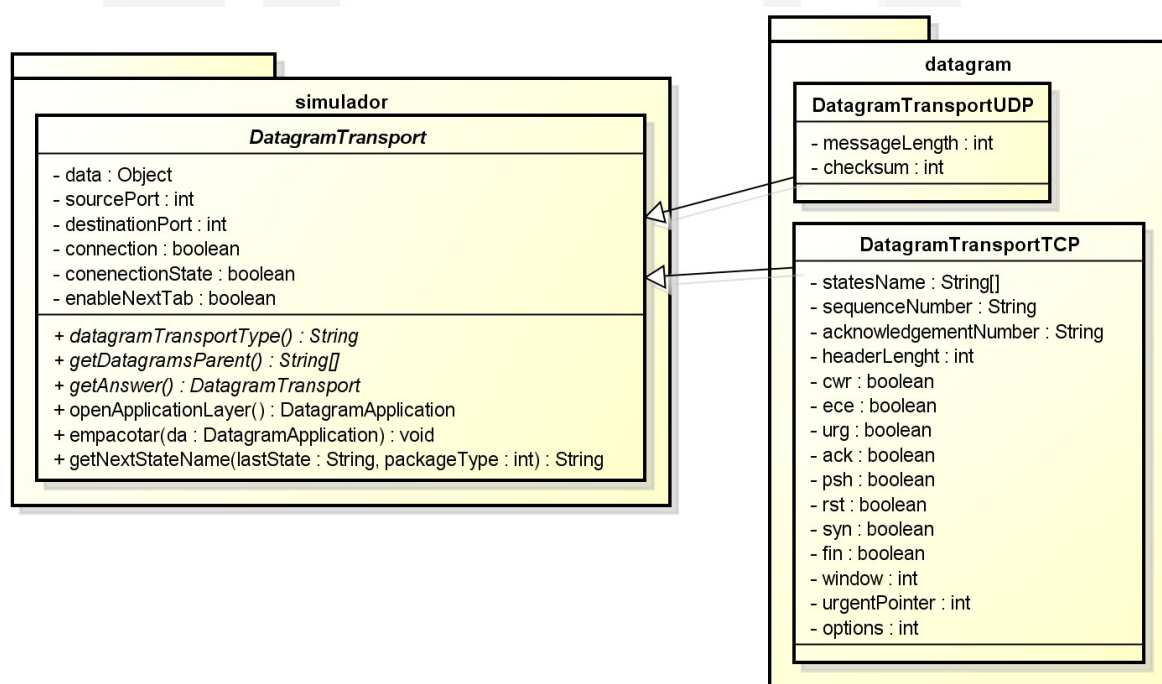
Esta divisão foi realizada para facilitar o carregamento dinâmico dos protocolos implementados no simulador. Este carregamento é melhor descrito na Seção 4.3.5.

4.2.1 Classes Datagram

As classes **Datagram** contém os dados do cabeçalho de cada um dos protocolos. Elas são responsáveis pelo controle lógico do funcionamento dos protocolos, definindo suas principais características. Para cada camada do modelo TCP/IP, existe uma classe **Datagram** abstrata. Por exemplo a classe **DatagramTransport** se refere a camada de transporte do modelo.

Cada um dos protocolos implementados devem estender uma classe **Datagram** abstrata. Na camada de transporte foram implementados dois protocolos, TCP e UDP, e para isso foram criadas duas classes **DatagramTransportTCP** e a **DatagramTransportUDP**, respectivamente, ambas estendendo a classe **DatagramTransport**. A relação entre as classes abstratas e suas filhas é demonstrada na Figura 18.

Figura 18 – Diagrama de Classes dos Datagrams

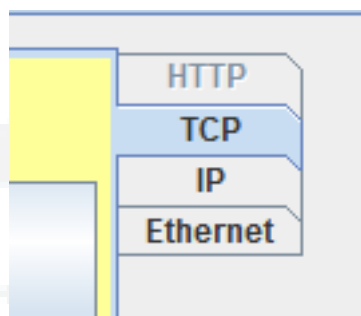


Fonte: Elaborado pelo autor.

Todas as classes filhas também devem implementar alguns métodos abstratos. Um destes métodos é o **getDatagramsParent()** que retorna um *array* de *strings* com o nome dos protocolos, da camada inferior, em que eles podem ser encapsulados. Por exemplo o **Datagram** que implementa o protocolo TCP retorna um *array* contendo a *string* IP pois o protocolo TCP é encapsulado no protocolo IP.

Além destes, existe um método responsável por retornar o nome do protocolo, sendo útil para a parte visual do simulador. No caso do protocolo TCP o método retornaria somente a *string* “TCP”. Este texto é utilizado nas abas dos diferentes protocolos, como demonstrado na Figura 19.

Figura 19 – Abas dos protocolos



Fonte: Elaborado pelo autor.

Outro método importante é o **getAutomaticAnswer()**, que retorna um **ArrayList** contendo objetos **Datagram** do mesmo tipo da classe pai. Esta função é necessária para que o simulador possa responder automaticamente um pacote, sem a intervenção do usuário. É utilizado um **ArrayList** devido ao fato de que em algumas circunstâncias ser necessário enviar mais de uma mensagem em resposta a uma pacote. Um exemplo é o encerramento de conexão do protocolo TCP, onde é necessário enviar um pacote com a *flag* ACK ativa em resposta ao pedido de finalização de conexão vinda do cliente, e logo em seguida enviar um outro pacote com a *flag* FIN ativa solicitando um encerramento da conexão. Este procedimento é necessário já que a conexão deve ser finalizada em ambas as partes, como descrito na Seção 2.4.2.

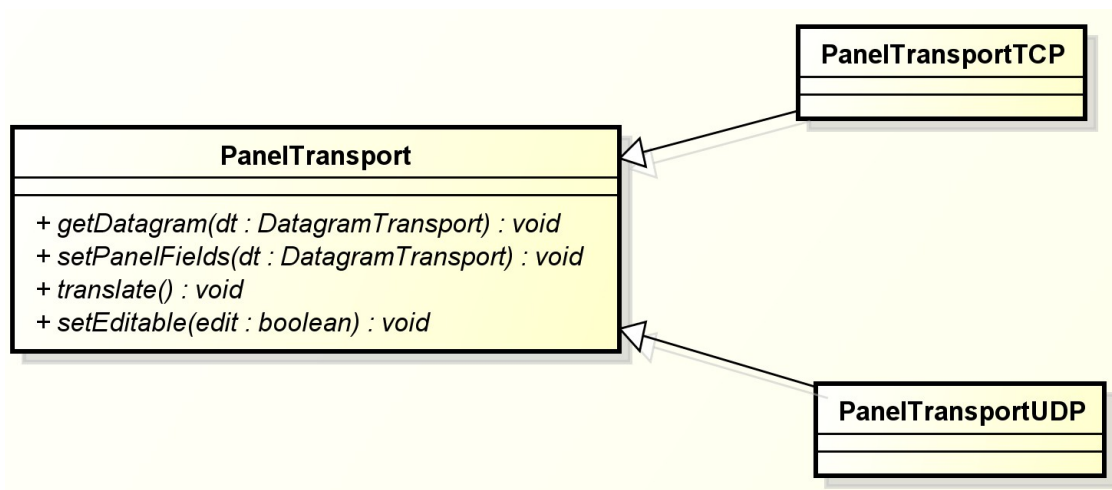
A nomenclatura da classe é algo importante para o carregamento dinâmico dos protocolos e essencial para o funcionamento correto da aplicação. O nome da classe deve iniciar com **Datagram**, seguido pelo nome da camada e o nome do protocolo.

4.2.2 Classes Panel

Estas classes são responsáveis pela parte visual dos protocolos, semelhante as classes **Datagram** existente uma classe abstrata para cada camada do modelo TCP/IP porém sua nomenclatura começa com **Panel**. Por exemplo a classe responsável pela camada de transporte é a classe **PanelTransport**, da mesma maneira a classe responsável pelo protocolos

TCP, que estende a classe **PanelTransport**, é nomeada **PanelTransportTCP**. A relação entre as classes abstratas e suas filhas é demonstrada na Figura 20.

Figura 20 – Diagrama de Classes dos Panels



Fonte: Elaborado pelo autor.

Para cada um dos campos do cabeçalho de um dos protocolos existe um componente de Swing para permitir alterar seu valor. Os valores dos campos são salvos em um objeto do tipo **Datagram**. Quando o usuário enviar o pacote criado para outra estação, os objetos **Datagrams** são serializados e enviados ao destinatário selecionado.

Estas classes também contêm métodos abstratos, importantes para o funcionamento da aplicação. Dois destes métodos são responsáveis por manipular os valores dos campos do cabeçalho de um protocolo. No caso, são os métodos **setDatagram()** e **setPanelFields()**, ambos recebendo um objeto do tipo **Datagram** como parâmetro. O primeiro método atualiza os dados do **Datagram** de acordo com os valores do **Panel** e o segundo preenche os campos do **Panel** de acordo com os dados do **Datagram**.

Outro método que cada classe filha deve implementar é o **translate()**, que é responsável pela internacionalização dos textos dos campos das classes **Panel**, obtendo as traduções a partir de um **ResourceBundle**.

Além desses, há método **setEditable()**, que é utilizado para definir se os campos dos painéis podem, ou não, ser editados. Ao se criar um novo pacote é habilitada a edição dos campos, porém ao visualizar um pacote recebido é permitido apenas visualizar os valor dos campos, não podendo alterá-los.

4.2.3 Classe WorkStation

A classe **WorkStation** guarda as informações sobre as estações adicionadas no mapa de rede. Os principais atributos dela são o IP, o número MAC, o *nickname* e a porta de conexão. As informações de IP e porta são essenciais pois é com estas informações que é estabelecido a conexão, via *sockets*, com a outra máquina.

4.2.4 Classe Packet

A classe **Packet** guarda as informações dos pacotes enviados e recebidos. Como a estação de origem, a estação de destino e o *status* do pacote recebido. Os pacotes enviados não tem *status*. Os *status* possíveis para os pacotes são:

- UNREAD: pacote recebido e não lido;
- READ: pacote recebido e já lido;
- DROPPED: pacote descartado.

Nela também estão contidos quatro objetos do tipo **Datagram**, um para cada camada da modelo TCP. Todas as informações são serializadas sob demanda, para poderem ser enviadas para as demais estações.

4.2.5 Classe Netstat

A classe **Netstat** é responsável pelo controle de cada uma das portas dos protocolos da camada de transporte, que estabelecem conexão. Além disto possibilita a visualização, de forma gráfica, dos estados de cada uma das portas ativas.

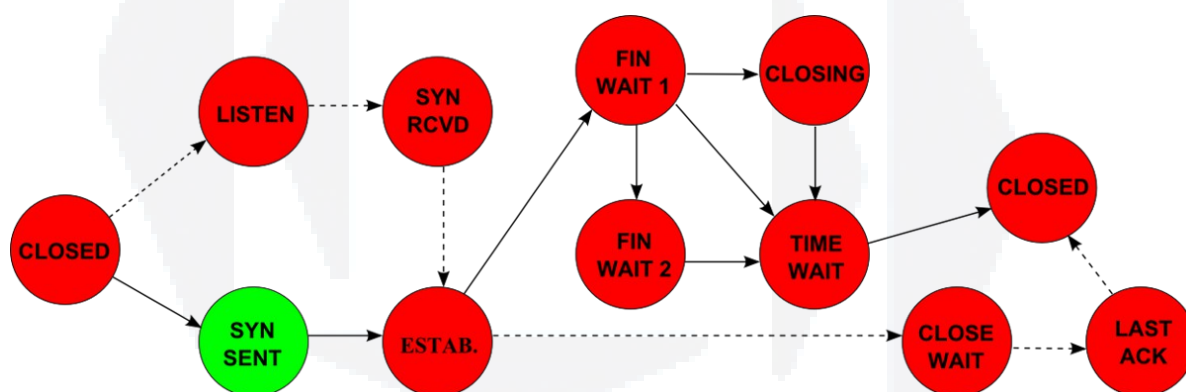
Para realizar o controle das portas foi utilizado um **HashMap**, onde a chave representa a porta do endereço local e o valor é um **ArrayList** contendo as demais informações sobre a conexão - como o endereço externo (IP e número da porta), nome do protocolo e estado atual da porta. Estas informações são apresentadas para o usuário por meio de um componente **JTable**.

Nesta classe existem dois métodos que manipulam os estados das portas: o método **validateRecicedPacket()** e o **validateSentPacket()**, sendo executados quando um pacote for recebido e enviado, respectivamente. Ambos os métodos recebem um objeto **Packet** e

retornam um *boolean* informando se o pacote é válido ou não. Além disso eles alternam o *status* das portas e quando a conexão for encerrada, liberam a porta.

O método responsável por mostrar, de forma gráfica, os estados de uma determinada conexão é o **createStates()**. Este método verifica a porta selecionada no **JTable** e exibe uma imagem da máquina de estados do TCP onde o estado atual da conexão da porta selecionada aparece de forma destacada. Um exemplo desta imagem por ser vista na Figura 21, onde aparece o estado SYN SENT destacado. Foi criada uma imagem para cada um dos estados desse protocolo. Para facilitar a implantação de outros protocolos com estados, as imagens devem ser nomeadas segundo o formato **nome_do_estado.png** e ficarem dentro de uma pasta com o nome de protocolo. Neste caso, tanto o nome da pasta quanto o do arquivo devem ser escritos em letras minúsculas.

Figura 21 – Diagrama de estados do protocolo TCP



Fonte: Elaborado pelo autor.

Outro método importante para o mecanismo de resposta automática, que será abordado na Seção 4.3.1, é o **findFreePort()**. Sua função é procurar uma porta livre, ou seja sem nenhuma conexão ativa. Foi definido um intervalo de busca entre as portas de número 6000 até a porta de número 65535.

4.2.6 Classe ArpTable

Esta classe cria uma tabela relacionando o MAC e o IP de cada uma das Workstations que estão no mapa de rede. É utilizado um **ArrayList** para guardar a lista de IPs e um **JTable** para mostrar a tabela ARP, onde em uma coluna fica o valor do IP e da porta de conexão e em outra o valor do número MAC da respectiva estação.

4.2.7 Classe FindStation

Esta classe utiliza *sockets*, enviando mensagens em *multicast*, para encontrar outros simuladores na rede local. Cada mensagem contém um objeto **Workstation**, com as configurações locais da estação, que é serializado e enviado utilizando uma *thread*, para que não trave a aplicação. O envio é feito através da classe **MulticastSocket**, para o IP *multicast* 228.5.6.7 na porta 9000.

Em cada uma das estações está rodando uma outra *thread*, que fica esperando uma conexão no mesmo endereço IP. Quando uma estação recebe o objeto **Workstation** enviado, ela retorna para o remetente um outro objeto **Workstation** contendo suas configurações porém desta vez utilizando um envio *unicast*.

4.3 Modificações realizadas

Nesta seção serão detalhadas as principais modificações e melhorias realizadas no simulador desenvolvido por Müller(2010) com o intuito de facilitar a interatividade do usuário com a aplicação e demonstrar outras funcionalidades do modelo TCP/IP não tratadas anteriormente.

4.3.1 Resposta Automática de Pacotes

Foi desenvolvido um algoritmo que responde de forma automática a alguns pacotes recebidos. Como exemplo pode-se citar algumas requisições ICMP, solicitações de estabelecimento e encerramento de conexão TCP e requisições de GET do HTTP.

O controle da resposta automática é feita através de uma *flag* que por padrão está desabilitada. Quando habilitada o simulador cria e envia pacotes destinados à estação que lhe enviou um pacote, sem que o usuário tenha que fazer nenhuma ação.

As respostas são criadas pelas classes **Datagram** de cada um dos protocolos que constituem o pacote. Através do método **getAutomaticAnswer()**, que analisa os valores dos campos do **Datagram** recebido e retorna um outro objeto **Datagram** com uma resposta equivalente. Os objetos **Datagram** dos protocolos da camada de transporte ainda levam em conta o estado atual do conexão para definir a resposta.

Caso todos os objetos **Datagram** tenham uma resposta válida, o simulador envia uma resposta contendo cada um destes **Datagram** encapsulados. Se um destes **Datagram** não tiver nenhuma resposta válida, então nenhuma resposta é enviada.

4.3.2 Criação de estação secundária

Para permitir que duas instâncias do simulador possam ser executadas ao mesmo tempo em uma mesma máquina, foram definidas duas portas distintas para o estabelecimento de conexão entre as estações, a porta 7000 e a 8000.

Sempre que o simulador é executado, é feita uma tentativa de conexão utilizando a porta 7000. Caso ocorra uma exceção no estabelecimento da conexão, uma mensagem é mostrada ao usuário avisando que já existe uma instância do simulador sendo executada e a conexão é feita utilizando a porta 8000.

Não é possível ter mais que duas instâncias do simulador em uma mesma máquina. Ao tentar criar outra estação uma mensagem é exibida avisando que não é possível executar outro simulador.

4.3.3 Criação do pacote ICMP

Foi implementado um novo protocolo no simulador de redes, o protocolo ICMP. Para a criação deste novo protocolo foi necessário o desenvolvimento de duas novas classes, a classe **DatagramInternetICMP** e a classe **PanelInternetICMP**.

A classe **DatagramInternetICMP** é responsável pelo controle das informações contidas no cabeçalho do protocolo. Como o ICMP pertence à camada de redes mas é encapsulado dentro de um pacote IP, decidiu-se que a classe **DatagramInternetICMP** seria uma especialização da classe **DatagramInternetIP**. Desta forma os pacotes ICMP teriam todas as características de um pacote IP e mais as características do ICMP que foram implementadas.

A outra classe, a **PanelInternetICMP**, é responsável pela parte visual do pacote. Para poder distinguir os campos referentes ao protocolo IP dos campos de ICMP foi feita uma divisão em dois painéis distintos.

Foram encontradas algumas dificuldades para a criação do *layout* final do pacote do protocolo ICMP, por causa da estrutura definida previamente. Após algumas tentativas distintas para a melhor disposição das informações na tela, optou-se pelo *layout* demonstrado na Figura 22. É possível observar os dados referentes ao IP na parte superior do pacote, enquanto os dados referentes ao ICMP encontram-se na parte inferior do pacote.

Figura 22 – Pacote ICMP gerado pelo simulador

Simulador

Arquivo Visualizar Opções Ajuda

Detalhes do Pacote

Pacote IP

Bit Offset: 00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Versão: 4				IHL: 5				Serviços diferenciados: 255				Tamanho total: 65535																			
Identificação: 65535																IP		Deslocamento de fragmentos: 8191													
Tempo de Vida: 255								Protocolo: 4								Checksum do Cabeçalho: 65535															
Endereço de Origem: 10.3.18.40																															
Endereço de Destino: 10.3.18.37																															
Opções: 0																															

Pacote ICMP

Bit Offset: 00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Tipo: 0: Echo Reply																Código: 0: Echo reply															
Checksum																															
Dados ICMP																															

Descrição do Protocolo ICMP

um roteador, o envio é relatado ao transmissor pelo ICMP (Internet Control Message Protocol).

O ICMP também é usado para testar a Internet. Cerca de 12 tipos de mensagens ICMP são definidos. Cada tipo de mensagem ICMP é transportada encapsulada dentro de um pacote IP.

Enviar

MAC: 78-2B-CB-C0-6F-BF IP: 10.3.18.40 Porta: 7000 Nick: Estação 1 Resposta Automática Habilitada

Fonte: Elaborado pelo autor.

Como foi descrito na Seção 2.2.3, no protocolo ICMP o campo código depende do campo tipo e seus valores já são pré-definidos na estrutura do protocolo. Por esta razão, para editar os valores destes campos foi implementado dois componentes **JComboBox**.

Ao modificar o valor do campo tipo, um algoritmo modifica dinamicamente os possíveis valores do campo código, incluindo as opções de códigos que estão previstas no protocolo para o tipo selecionado.

4.3.4 Remoção da dependência da camada de aplicação

Na versão anterior do simulador o usuário era obrigado a criar um pacote contendo um protocolo de cada uma das camadas do modelo TCP/IP para poder enviá-lo a outra estação.

Esta dependência impediria o funcionamento correto de um pacote ICMP, por exemplo. Tal protocolo, que está contido na camada de rede, não contém nenhum outro pacote encapsulado nele, tornando as camadas de transporte e aplicação desnecessárias para este caso.

Para solucionar este problema, o código fonte original foi alterado. Prevendo a possibilidade da criação e envio de objetos das classes **Datagram** com valor nulo. Assim somente as camadas dos protocolos selecionados serão criadas, as demais ficarão ocultas.

4.3.5 Criação de algoritmo de carregamento dinâmico de objetos Datagram

Para a seleção dos protocolos de cada uma das camadas do modelo TCP/IP, que são utilizados para o criação de um pacote, é utilizado o componente **JComboBox**. Anteriormente, os itens contidos nestes elementos eram definidos estaticamente.

Nesse contexto, caso fosse criado um novo protocolo, como o ICMP, seria necessário mudar o código fonte original e acrescentar esta nova opção manualmente. Para facilitar que sejam criados novos protocolos de forma mais dinâmica foi desenvolvido um algoritmo que “varre” todos as classes da aplicação e preenche o componente **JComboBox** automaticamente.

Inicialmente, foi modificada a estrutura das classes do software original, criando um novo *package* denominado **datagram**. Nele foram postas todas as classes não abstratas do tipo **Datagram** que contém cada um dos protocolos implementados.

Após foi criado o algoritmo de varredura que é dividido em duas partes. Uma parte é responsável por verificar as classes no software já compilado. Outra parte analisa os arquivos quando o programa é executado diretamente no ambiente de desenvolvimento. Dependendo da situação o algoritmo executa uma ou outra parte. Ambas as partes varrem somente os arquivos contidos no *package* **datagram**.

Tendo encontrado todas as classes **Datagram** não abstratas disponíveis, as mesmas são divididas em quatro *arrays* distintos, um para cada camada do modelo TCP/IP. Para

descobrir de qual camada pertence cada uma das classes é analisada a classe que o **Datagram** estende. Por exemplo, a classe **DatagramTransportTCP** estende o classe **DatagramTransport**, logo a mesma pertence à camada de transporte.

Em cada uma das classes abstratas do tipo **Datagram**, que são responsáveis pela estrutura do pacote, foi criado o método abstrato **getDatagramsParent()**. Este método retorna um *array* de *strings*, em que cada um dos valores corresponde ao nome de um dos protocolos em que este pacote pode ser encapsulado.

Ao selecionar um protocolo da camada de enlace, por exemplo, são carregados no **JComboBox** da camada de rede todos os protocolos da camada superior que podem ser encapsulados no protocolo selecionado. Esta mesma lógica é utilizada para todas os protocolo das demais camadas do modelo TCP/IP implementados.

5 ANÁLISE DA APLICAÇÃO

Neste capítulo serão descritos os principais recursos que o simulador oferece, demonstrando sua interface gráfica e as formas de interação com o usuário.

5.1 Fluxo de Telas

Como o simulador tem como principal objetivo ser uma ferramenta didática, as interfaces foram criadas para ter uma simples interação com o usuário, exibindo apenas as informações essenciais para sua utilização.

Esta seção irá descrever as principais decisões na implantação das interfaces e as formas como o usuário pode interagir com a aplicação.

5.1.1 Rodapé

No rodapé da aplicação estão as informações sobre a estação. Dentre elas estão o número MAC e IP, que são os mesmos da placa de rede do computador. Além da porta de conexão, onde a estação primária utiliza a porta 7000 e a secundária a 8000, e o *nickname* da estação. O rodapé da aplicação é mostrado na Figura 23.

Figura 23 – Rodapé do simulador

MAC: 78-2B-CB-C0-6F-BF IP: 10.3.18.40 Porta: 7000 Nick: Estação 1 Resposta Automática Habilitada

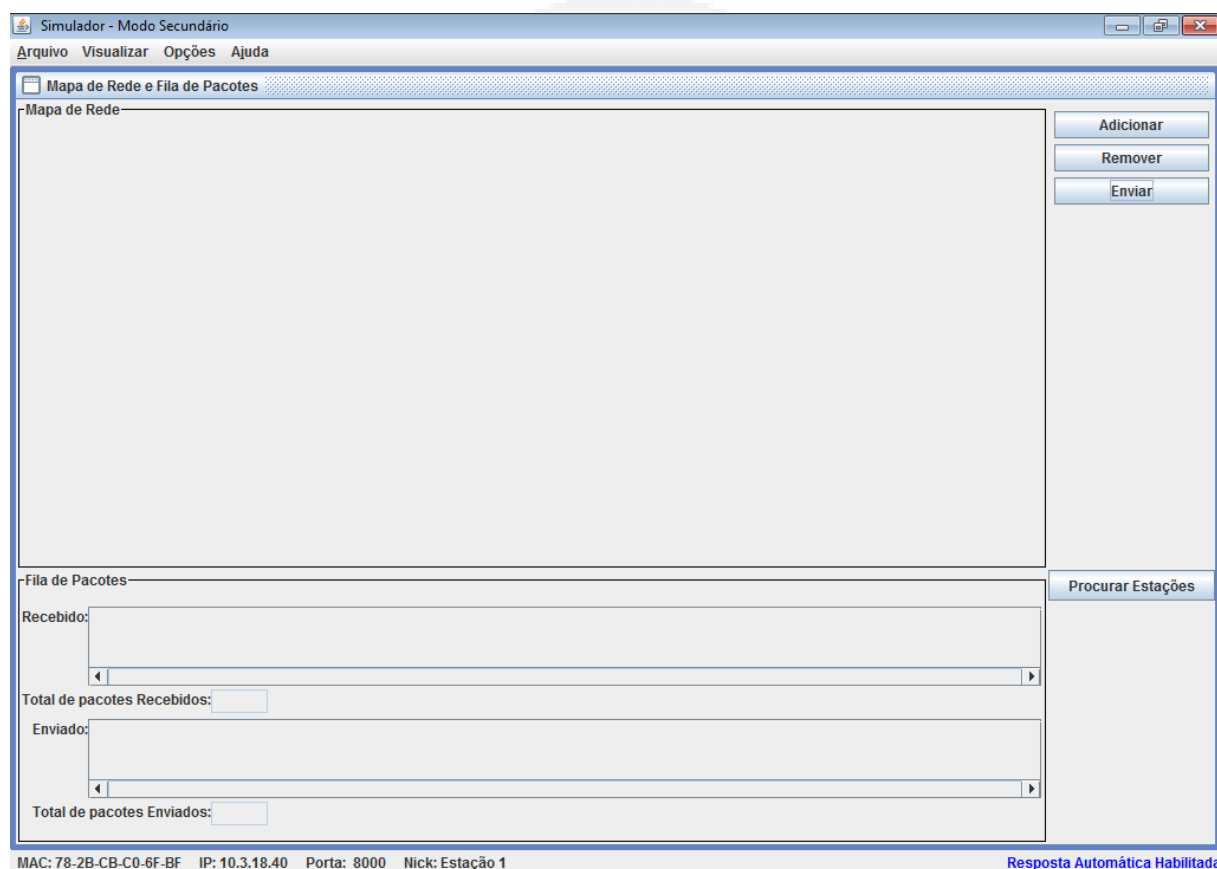
Fonte: Elaborado pelo autor.

No rodapé também é mostrado se a autoresposta está ativa ou não, utilizando uma mensagem com fonte de cor azul quando habilitada e em vermelho quando desabilitada.

5.1.2 Tela Principal

Para facilitar a utilização do aplicativo as principais informações sobre o simulador estão contidas na tela principal. Ela pode ser dividida em três partes distintas: o mapa de redes, a fila de pacotes e os botões de interação. A Figura 24 mostra a tela principal ao iniciar o simulador, com o mapa de rede e as filas de pacotes vazias.

Figura 24 – Estado inicial da Tela Principal



Fonte: Elaborado pelo autor.

O mapa de redes mostra todas as estações que fazem parte da simulação. São para estas máquinas que podem ser enviados os pacotes construídos no simulador. São exibidas as informações de IP, *nickname* e porta de conexão de cada uma das estações.

A fila de pacotes é dividida em duas partes os pacotes enviados e os pacotes recebidos. Existem basicamente quatro tipos de pacotes (os pacotes enviados, os recebidos e não lidos, os recebidos e lidos e os recebidos e descartados), cada um com um ícone distinto. É possível visualizar ambos os tipos de pacotes, para isso basta dar um duplo clique com o botão

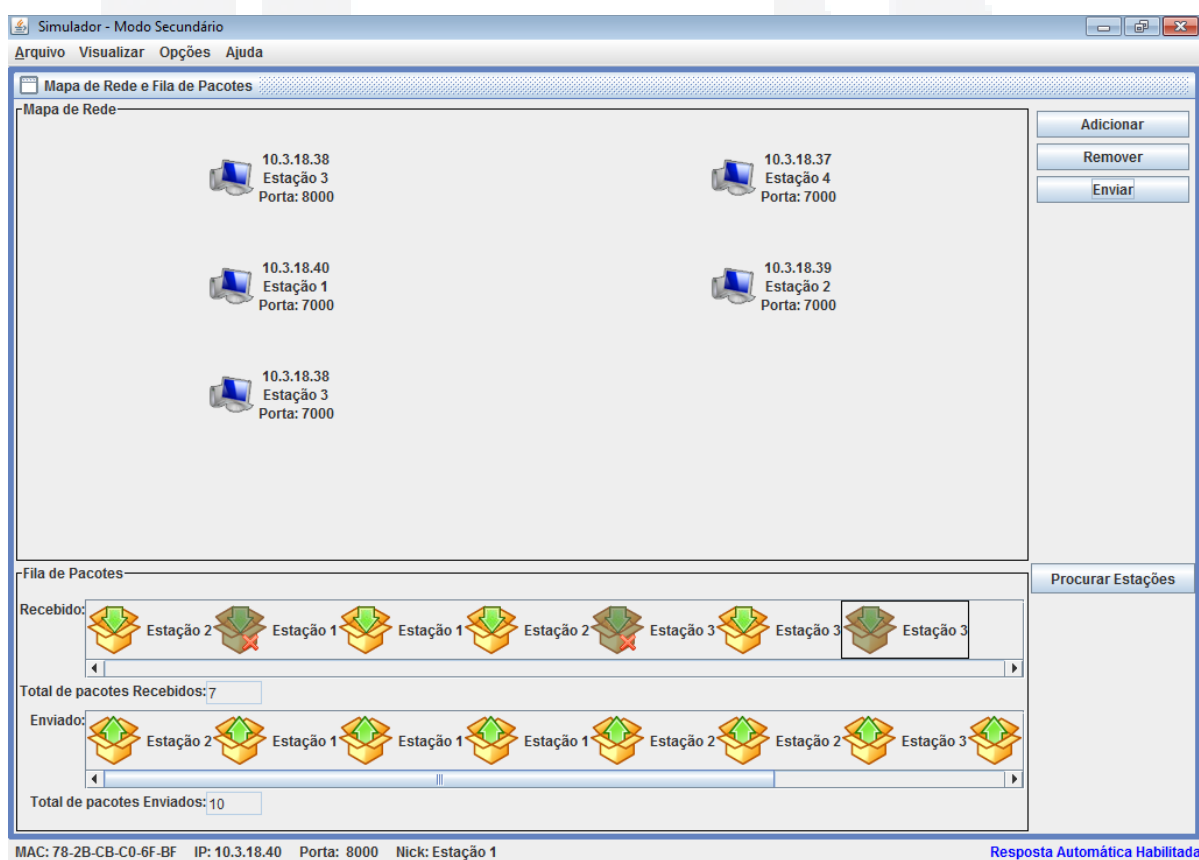
esquerdo do mouse, que será aberta a tela de visualização do pacote, porém com permissões diferentes.

Nesta tela ainda existem quatro botões para as principais interações do usuário. Os botões “Adicionar” e “Remover” são utilizados para gerenciar manualmente as estações do simulador. O botão “Adicionar” inclui uma nova estação com base em um IP informado pelo usuário e o botão “Remover” remove a estação selecionada.

Ainda existe o botão “Enviar” que, havendo pelo menos uma estação incluída, abre a tela de criação de pacotes além do botão “Procurar Estações” que executa a busca automática de novas estações.

Após a busca das estações contida na rede e da troca de alguns pacotes entre as estações, a fila de pacotes e o mapa de redes são populados. A Figura 25 demonstra a tela principal depois destas interações.

Figura 25 – Tela Principal após interações

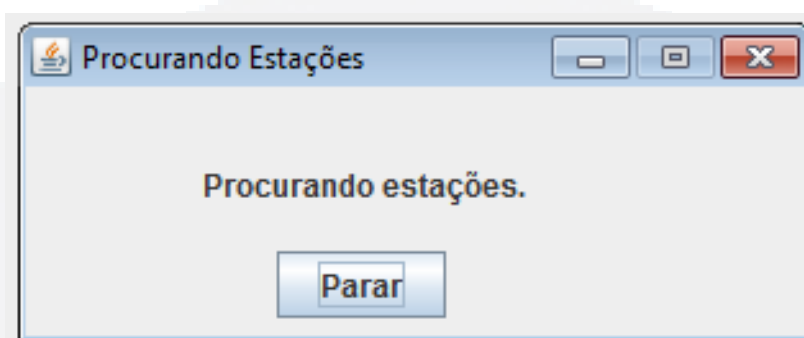


Fonte: Elaborado pelo autor.

5.1.3 Busca de novas estações

Ao clicar no botão “Procurar Estações” é aberta uma janela (Figura 26), por cima da tela atual, com uma mensagem avisando que está sendo feita a procura de novas estações. Esta janela não bloqueia a tela principal e é fechada somente quando o usuário desejar, finalizando assim a busca. A mensagem tem uma animação para reforçar que a busca está sendo realizada.

Figura 26 – Tela de busca de novas estações



Fonte: Elaborado pelo autor.

5.1.4 Criação de pacotes

Na tela de criação de pacotes é possível escolher o destinatário para qual se deseja enviar o pacote e os protocolos de cada uma das camadas do modelo TCP/IP que irão compor o pacote. Ao escolher um dos protocolos da camada de enlace, são liberados todos os protocolos da camada internet que podem ser encapsulados dentro deles, e assim ocorre para as demais camadas.

Caso o protocolo selecionado transporte estruturas da camada superior, como o IP ou o TCP, o usuário é obrigado a escolher um protocolo desta outra camada. Neste caso o botão de criação do pacote é desativado e uma mensagem é exibida ao usuário informando a necessidade da escolha do protocolo.

Se o protocolo selecionado não transporte estruturas da camada superior, como o ICMP ou o UDP, então o botão de criação do pacote fica habilitado. Caso o usuário deseje criar o pacote ele será direcionado para a tela de visualização de pacotes.

A tela de criação de pacotes é mostrada na Figura 27.

Figura 27 – Tela de criação de pacotes

A imagem mostra uma janela de diálogo intitulada "Criar Pacote". Ela contém quatro campos de seleção: "Destino:" com o valor "192.168.1.101:8000 - Cristiano", "Enlace:" com o valor "Ethernet", "Internet:" com o valor "IP" e "Transporte:" que está vazio. Abaixo desses campos, há uma mensagem em vermelho: "Escolha o protocolo da camada de Transporte!". Na base da janela, há dois botões: "Cancelar" e "Criar".

Fonte: Elaborado pelo autor.

5.1.5 Visualização de pacotes

Na tela de visualização de pacotes é possível visualizar e manipular os campos do cabeçalho e de dados de cada um dos protocolos que fazem parte do pacote construído. Na lateral estão localizadas abas, com o nome do protocolo, que serve para alternar entre os protocolos que compõem o pacote. Esta alternância também pode ser feita através do campo de dados de um protocolo.

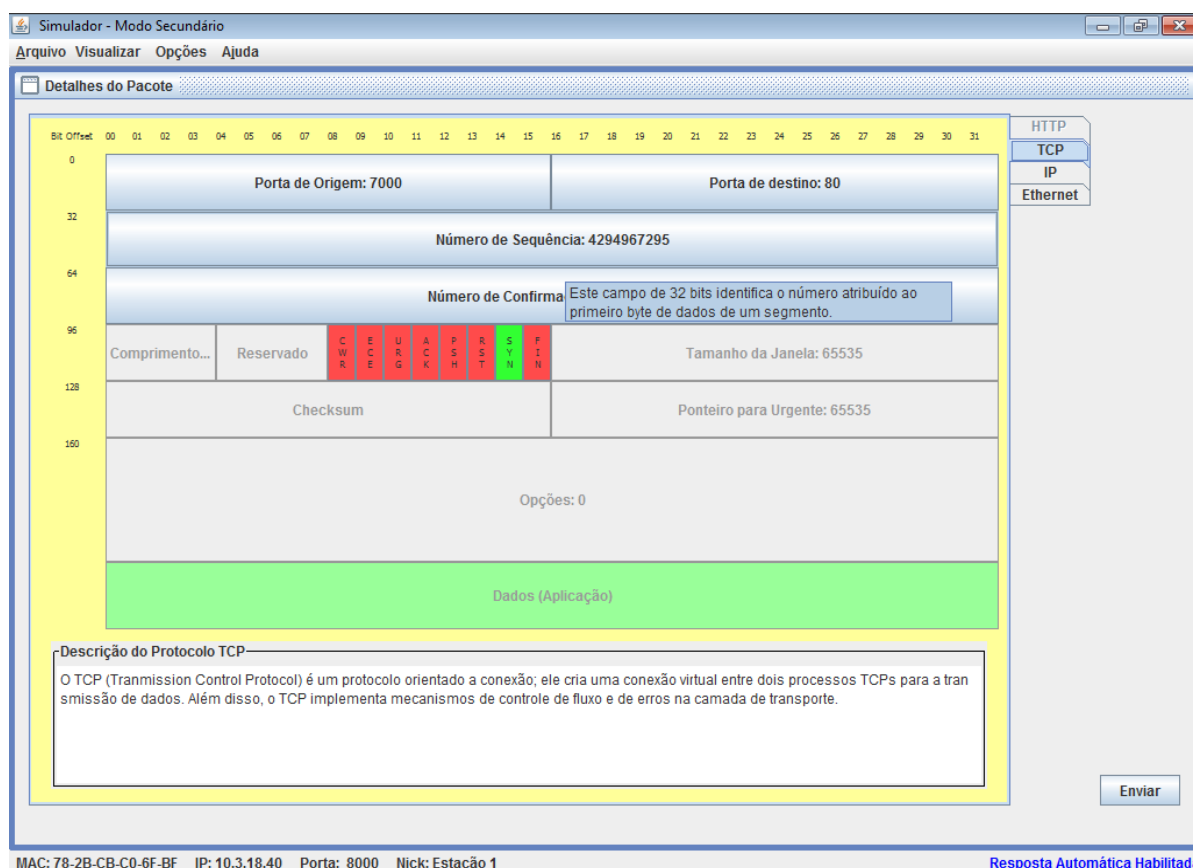
Em alguns casos não é possível modificar os valores de um protocolo superior, como por exemplo o caso do HTTP. Para poder enviar uma mensagem HTTP é necessário que, anteriormente, seja estabelecida uma conexão TCP e só depois disso o acesso a camada de aplicação é liberada.

Existem três formas possíveis de se visualizar um pacote. Ao criar um novo pacote ou visualizar um pacote enviado, os campos ficam liberados para edição, podendo alterar os valores da maioria dos campos dos protocolos. E caso o usuário desejar este pacote pode ser enviado ao destinatário previamente escolhido.

Se o usuário visualizar um pacote recebido não descartado todos os campos estarão bloqueado para edição, sendo possível somente visualizar seus valores. Neste caso existe a opção de responder o pacote. Ao escolher a opção de responder um novo pacote é criado, com

o valor de alguns campos, como o endereço de origem e o endereço de destino no protocolo IP, invertidos e o destinatário é definido como o remetente do pacote anterior. Após a criação do pacote os campos são liberados para edição e a opção de enviar o pacote é liberada. A tela de visualização dos pacotes, com os campos liberados para edição, é mostrada na Figura 28.

Figura 28 – Tela de visualização de pacotes



Fonte: Elaborado pelo autor.

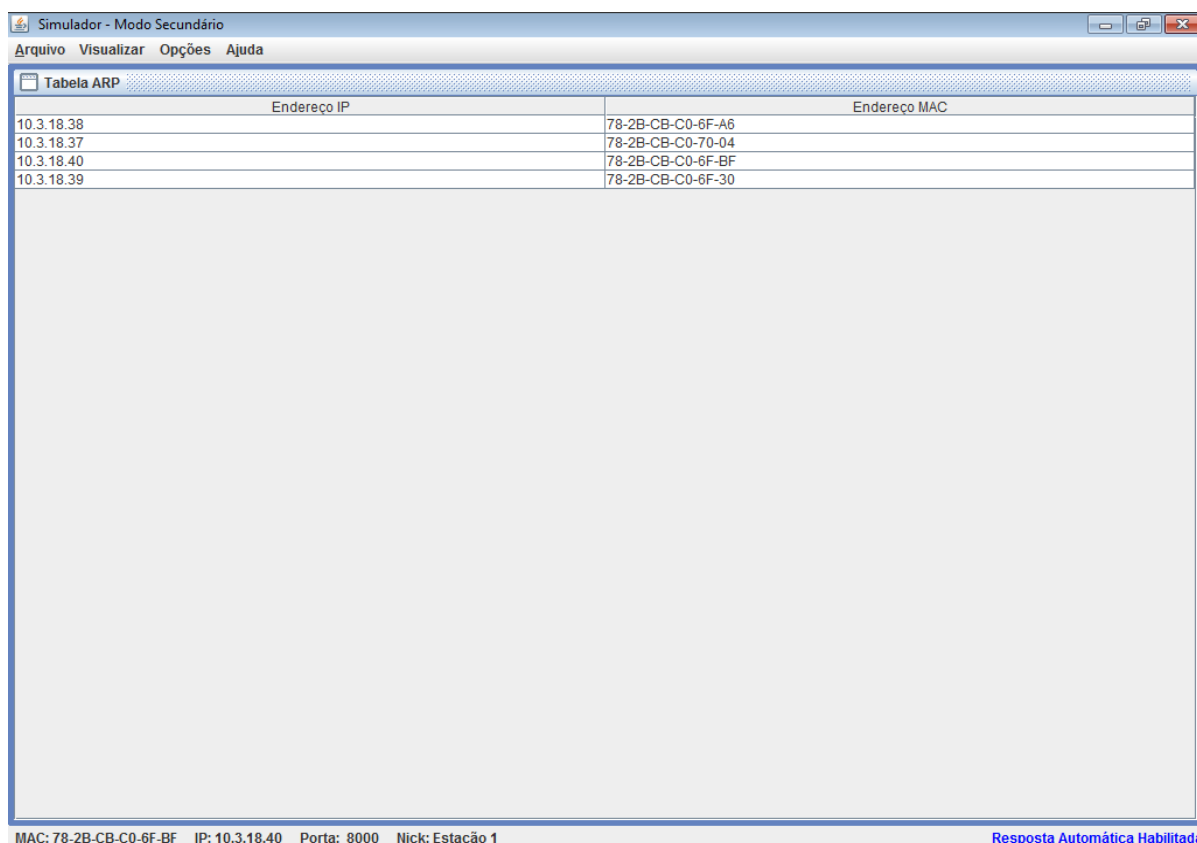
Quando um pacote descartado é visualizado todas as opções são desabilitadas, não sendo possível nem modificar o valor dos campos nem responder este pacote. É possível somente visualizar seus valores.

Em cada um dos protocolos implementados tem uma área de texto, na parte inferior da tela de visualização, onde tem uma pequena descrição sobre o protocolo demonstrado. Além disso cada um dos campos têm uma descrição que é visualizada, na forma de um *tooltip*, ao passar o mouse por cima do mesmo. Para alterar as configurações do *tooltip*, como o tempo de exibição e a largura da mensagem, foi criada uma nova classe, denominada **ToolTipButton**, que estende a classe **Jbutton**.

5.1.6 Tabela ARP

Esta tela lista a tabela ARP relacionando o IP com o número MAC de cada uma das estações de contato. Com estas informações é possível demonstrar os conceitos de endereços físicos e endereços de rede. A tela da tabela ARP é mostrada na Figura 29.

Figura 29 – Tela Tabela ARP



Endereço IP	Endereço MAC
10.3.18.38	78-2B-CB-C0-6F-A6
10.3.18.37	78-2B-CB-C0-70-04
10.3.18.40	78-2B-CB-C0-6F-BF
10.3.18.39	78-2B-CB-C0-6F-30

Fonte: Elaborado pelo autor.

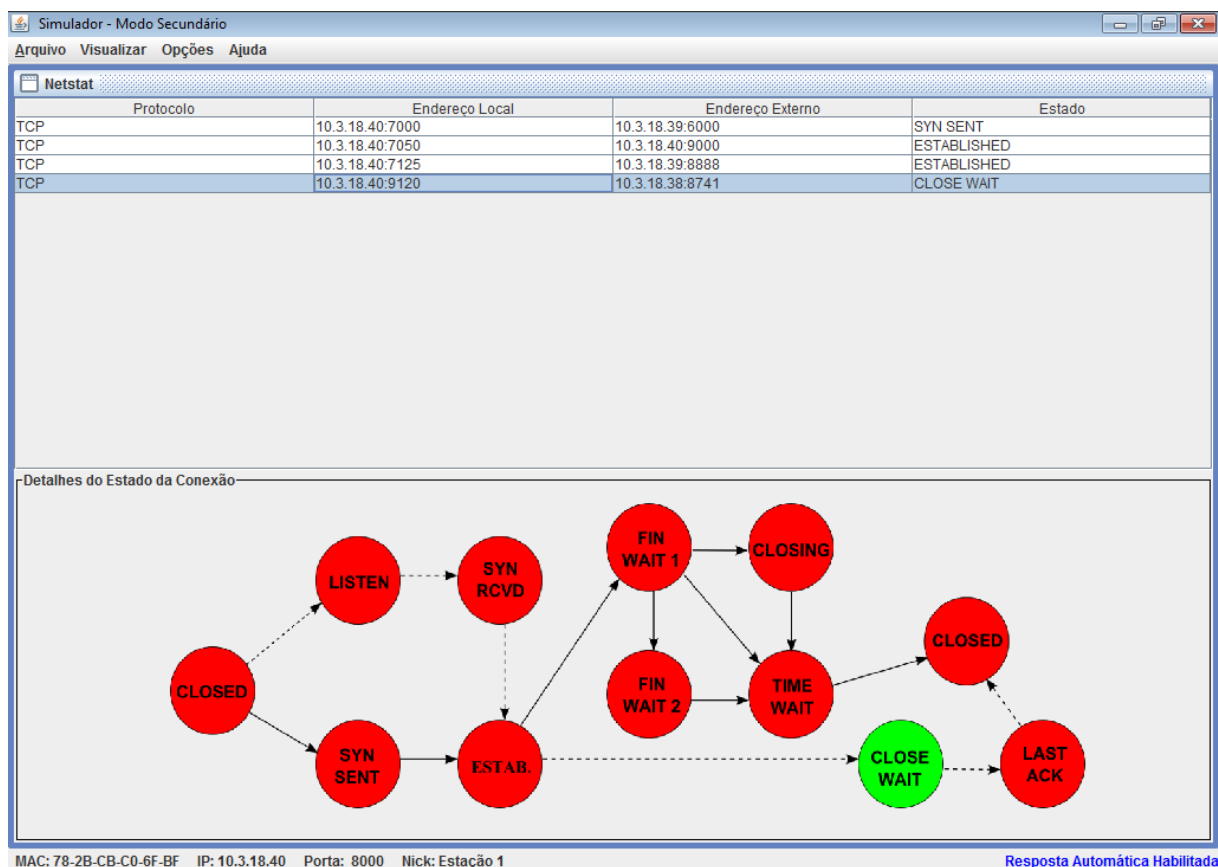
5.1.7 Netstat

O nome Netstat é baseado no comando *netstat*, comum em muitos sistemas operacionais, utilizado para se obter informações sobre as conexões da rede, tabelas de roteamento, entre outras informações da interface de rede.

Na tela Netstat, demonstrada na Figura 30, são apresentados os estados da conexão das portas ativas do simulador, além de mostrados, de forma gráfica, os estados do protocolo. Como o único protocolo implementado que apresenta estados é o TCP, somente as

informações de pacotes que utilizam este protocolo na camada de transporte serão mostradas nesta tela.

Figura 30 – Tela Netstat



Fonte: Elaborado pelo autor.

Esta tela é dividida em duas partes. Na parte de cima há uma tabela, dividida em quatro colunas, onde cada linha mostra a situação de uma das portas ativas. A primeira coluna mostra o nome do protocolo que está utilizando a porta, a segunda mostra o IP e porta utilizadas localmente nesta conexão, a terceira mostra o IP e porta do outro lado da conexão e a quarta coluna mostra o estado atual da porta.

Na parte de baixo é mostrada, através de uma imagem, todos os estados do protocolo da linha selecionada na tabela. Nesta imagem fica destacada, em outra cor, o estado atual da porta. Para cada um dos estados do TCP existem uma imagem com um estado diferente destacado.

Se o simulador receber ou enviar um novo pacote e este pacote estabelecer conexão em uma porta, a linha referente a esta porta é automaticamente selecionada e é mostrada a

imagem do estado atual desta porta. A qualquer momento o usuário pode selecionar outra linha, alterando também a imagem dos estados.

Quando for encerrada a conexão em uma das portas, esta porta é liberada e a linha, na tabela, referente a esta porta é removida e outra linha é selecionada.

5.1.8 Menu

Para ter acesso aos principais recursos e configurações do simulador foi criado um menu, na parte superior da aplicação. A Tabela 5 descreve os menus implementados, juntamente com suas opções e uma breve explicação sobre suas funcionalidades.

Tabela 5 – Menus da aplicação

Menu	Opções	Descrição
Arquivo	Sair	Finaliza a aplicação.
Visualizar	Mapa de Rede e Fila de Pacotes	Visualiza a tela de mapa de redes e a fila de pacotes.
	Detalhe de Pacote	Visualiza a tela de detalhe do pacote.
	Tabela ARP	Visualiza a tela da tabela ARP.
	Netstat	Visualiza a tela do netstat.
Opções	Auto Resposta	Ativa ou desativa a opção de auto resposta.
	Idioma	Altera (entre Português e Inglês) o idioma do simulador.
Ajuda	Sobre	Mostra informações do software.

Fonte: Elaborado pelo autor.

Ao clicar no menu “Visualizar” o usuário terá acesso as todas as telas do sistemas, este menu fica ativo durante toda a execução sendo possível alternar entre as telas a qualquer momento.

As opções do menu “Opções” também podem ser alteradas a qualquer momento. As configurações alteradas neste menu são salvas automaticamente ao fechar a aplicação e carregadas ao iniciá-lo novamente.

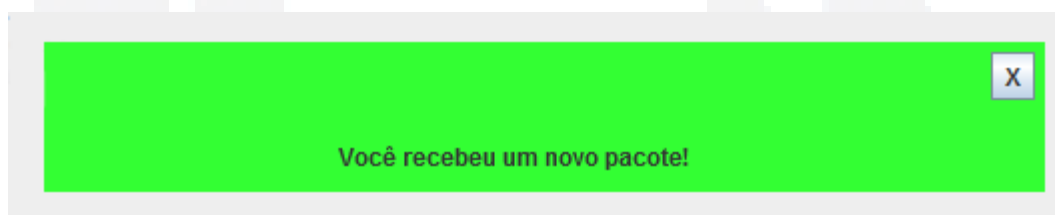
5.2 Mensagens para o Usuário

Anteriormente eram utilizados componentes do tipo **JOptionPane** para transmitir mensagens para o usuário. Porém o **JOptionPane** bloqueia a aplicação impossibilitando que o usuário faça qualquer outra ação sem antes fechá-lo.

Por esse motivo foi alterada a forma de exibição das mensagens sendo criada uma nova classe, que estende a classe **JFrame**, nomeada **SimulatorNotification**. Estas novas mensagens aparecem por cima da aplicação, centralizadas na tela, porém não bloqueiam a interação de usuário e são fechadas de forma automática após um determinado tempo ou quando o usuário desejar.

Existem dois tipos de mensagem: as mensagens de sucesso com o fundo verde e as mensagens de erro com o fundo vermelho. Um exemplo de uma mensagem do simulador e mostrada na Figura 31.

Figura 31 – Mensagem de sucesso do simulador



Fonte: Elaborado pelo autor.

5.3 Internacionalização

Para realizar a internacionalização foi utilizado a classe **ResourceBundle**. Esta classe lê arquivos com extensão **.properties** para obter os textos que serão exibidos na aplicação. Para cada *string* diferente existe uma chave relacionada, é através desta chave que é possível obter a *string*.

Foram criados dois arquivos diferentes, um para a língua portuguesa e outro para língua inglesa. Ao iniciar o simulador pela primeira vez o idioma definido é o português, sendo possível alterá-lo ao qualquer momento, em tempo de execução, através do menu da aplicação.

A maioria dos textos presentes na aplicação são traduzidos, como o título dos botões e menus, nome e descrição dos campos dos protocolos, mensagens para o usuário, entre outros.

As únicas informações que não são internacionalizadas são aquelas informadas pelo próprio usuário e algumas expressões em inglês como *checksum*, que não são alteradas mesmo quando for escolhido o idioma português.



6 CONCLUSÃO

Este trabalho apresentou a implementação de aprimoramentos um simulador didático de redes anteriormente desenvolvido por Müller(2010).

Entre as funcionalidades desenvolvidas está o controle e visualização dos estados do protocolo TCP, que permite demonstrar de forma gráfica os passos para estabelecimento e encerramento de conexão, auxiliando no ensino destes processos que muitas vezes pode ser difícil de compreender.

Além disso foi criado um mecanismo de resposta automática de alguns pacotes, que pode ser útil para simular a estrutura de cliente-servidor, onde um simulador assumiria o papel de servidor, respondendo de forma automática, e as demais estações enviariam mensagens para ele.

Para facilitar o uso do simulador em um ambiente doméstico, onde não há acesso a uma rede física nem a uma máquina virtual, foi permitido que sejam executadas duas instâncias do simulador numa mesma máquina, permitindo assim que seja feita a troca de mensagens entre simuladores em um único computador.

Analisando os resultado obtidos conclui-se que o simulador apresentado cumpre todos objetivos propostos, porém ainda existem funcionalidade que podem ser adicionadas futuramente. O controle de fluxo do modelo TCP é um exemplo, pois este é um assunto complexo e difícil de ser visualizado. Além disso poderia ser feito um controle melhor dos campos dos protocolos, pois hoje não é validado o tamanho nem o tipo dos dados inseridos. Outra implementação possível seria de um controle de *logs* do simulador, salvando as principais interações e erros ocorridos na simulação.

REFERÊNCIAS

- CISCO. Cisco Packet Tracer, version 5.3.0.0088. Disponível em: <http://www.baixaki.com.br/download/packet-trace.htm>. Acesso em: 1 dez. 2012.
- COMER, D. E. **Interligação em redes com TCP/IP**: Princípios, protocolos e arquitetura. 3 ed. Rio de Janeiro: Editora Campus, 1998. ISBN 85-352-0270-6
- COMER, D. E. **Redes de computadores e Internet**: Abrange transmissão de dados, ligações inter-redes, Web e aplicações. 4 ed. Porto Alegre: Bookman, 2009. ISBN 978-85-60031-36-8
- FIELDING, R. et al. **Hypertext Transfer Protocol – HTTP/1.1**: RFC 2116, 1999. Disponível em: <http://tools.ietf.org/html/rfc2616>. Acesso em: 18 dez. 2012.
- GLOMOSIM. GLOMOSIM - Global Mobile Information System Simulator, version 2.03. Disponível em: <http://pcl.cs.ucla.edu/projects/glomosim/academic/download.html>. Acesso em: 1 dez. 2012.
- INFORMATION SCIENCES INSTITUTE UNIVERSITY OF SOUTHERN CALIFORNIA. **Internet Protocol**: RFC 791, 1981. Disponível em: <http://www.ietf.org/rfc/rfc791.txt>. Acesso em: 17 dez. 2012.
- INFORMATION SCIENCES INSTITUTE UNIVERSITY OF SOUTHERN CALIFORNIA. **Transmission Control Protocol**: RFC 793, 1981. Disponível em: <http://www.ietf.org/rfc/rfc793.txt>. Acesso em: 17 dez. 2012.
- JUNIOR, J. H. T. et al. **Redes de computadores**: Serviços, administração e segurança. São Paulo: Markon Books do Brasil Editora Ltda, 1999. ISBN 85-346-0957-8
- KUROSE, J.; ROSS, K. **Redes de computadores e a Internet**: Uma abordagem top-down. 5 ed. São Paulo: Pearson, 2011. ISBN 978-85-88639-97-3
- MÜLLER, D. **Simulador didático de redes de computadores**. 2010. Dissertação (graduação em Engenharia da Computação), Centro Universitário Univates, Lajeado, 2010.
- NCTUNS. NCTUns – Network Simulator and Emulator, version 6.0. Disponível em: <http://nsl.csie.nctu.edu.tw/nctuns.html>. Acesso em: 1 dez. 2012.

NS. Network Simulator, version 2. Disponível em: <<http://www.isi.edu/nsnam/ns/ns-build.html>>. Acesso em: 1 dez. 2012.

OPNET. OPNET - Network Simulator. Disponível em: <<http://www.opnet.com>>. Acesso em: 1 dez. 2012.

POSTEL, J. **Internet Control Message Protocol**: RFC 792, 1981. Disponível em: <<http://tools.ietf.org/html/rfc792>>. Acesso em: 17 dez. 2012.

POSTEL, J. **User Datagram Protocol**: RFC 768, 1980. Disponível em: <<http://www.ietf.org/rfc/rfc768.txt>>. Acesso em: 17 dez. 2012.

TANENBAUM, A. S.; WETHERALL, D. J. **Redes de computadores**. 5. ed São Paulo: Pearson, 2011. ISBN 978-85-7605-924-0