UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO
COORDENAÇÃO DE PÓS-GRADUAÇÃO EM INFORMÁTICA

DISSERTAÇÃO DE MESTRADO

ESPECIFICAÇÃO DE COMPONENTES PARA SIMULAÇÃO DE REDES TCP/IP

MARCUS VINÍCIUS DA SILVA WAGNER

CAMPINA GRANDE-PB, BRASIL
AGOSTO DE 2000

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO
COORDENAÇÃO DE PÓS-GRADUAÇÃO EM INFORMÁTICA

MARCUS VINÍCIUS DA SILVA WAGNER

ESPECIFICAÇÃO DE COMPONENTES PARA SIMULAÇÃO DE REDES TCP/IP

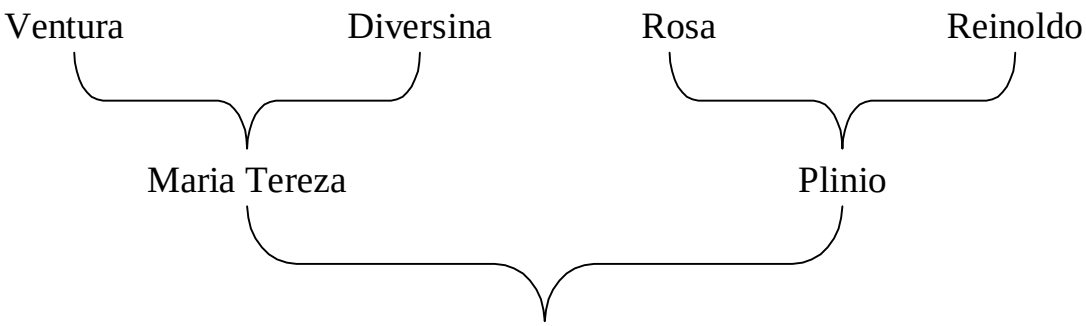
Dissertação submetida ao Curso de Pós-Graduação em Informática no Departamento de Sistemas e Computação do Centro de Ciências e Tecnologia da Universidade Federal da Paraíba, em cumprimento às exigências para obtenção do grau de Mestre em Informática.

Área de Concentração: Ciência da Computação
Sub-Área: Redes de Computadores

Orientadora: *Maria Izabel Cavalcanti Cabral*, D.Sc

Campina Grande, agosto de 2000.

DEDICATÓRIA



e...

a Deus.

AGRADECIMENTOS

Em primeiro lugar agradeço a minha orientadora Maria Izabel Cavalcanti Cabral que, com maestria, soube usar meus erros para mostrar o caminho do aprendizado, da qualidade de trabalho e sobretudo da importância do relacionamento humano. - Muito Obrigado Professora.

A Vera e a Aninha que com bom humor e categoria souberam conduzir momentos de nervosismo por parte dos alunos, demonstrando toda sua amizade.

Devo agradecer as instituições que estiveram diretamente ligadas a este meta alcançada: A UFPB pela estrutura física e humana, a CAPES pelos recursos finaceiros, e finalmente para UNIRONDON que investiu e incentivou a conclusão deste curso.

Ao prof. Mário (amigo que conheci no Tocantins e escreveu uma das cartas de recomendações de ingresso no mestrado) e aos professores do DSC que de uma forma ou de outra colaboraram para meu crescimento pessoal e intelectual, em especial deixo meu abraço para Arturo, Edilson, Peter, Marcus Sampaio, Jacques, Antão, Ulrich, Elmar, Bernardo, Homero, Marcelo Barros, Joberto, e a figura caricata do Agamenon Lopes. - Vocês são pontos de referência.

Neste período de Campina Grande tive o prazer de conhecer tantos amigos que nem sei contar. As baianas Pathy e Milena, Dani a potiguar, Ceni a maranhense, Cacá o paraense, Emy, Germana, Karine, Karina, Killer, Dalton, .. os Paraibanos de rocha, ..., tenho certeza nas cruzadas da vida vamos nos encontrar.

A Gauchada gremista Artur e Maurício e colorados Fabiano (que escreveu uma das cartas de recomendação ao meu ingresso no mestrado) e Sausen que já nos conhecíamos de Ijuí, sei que estaremos prontos para outras jornadas e vitórias do Grêmio.

Dilvan e Luiz Maurício, além de amigos também companheiros de "chatô", juntamente com Mauricio. Vocês foram fundamentais nesta caminhada.

Aos amigos de Cuiabá, Cristiano e Patrícia que acompanharam minha agonia e luta final em busca desta conquista, meu muito obrigado.

A minha amiga e colega Juliana pelas horas em que estudamos juntos na realização este trabalho.

A Naná que é minha mãe paraibana.

E não poderia deixar de agradecer ao alicerce desta conquista. - Minha família: mãe, pai e a Nãna.

Sei que estou esquecendo de muita gente, mas estou com os olhos cheios de lágrimas, e chegou o momento de levantar a cabeça e seguir meu caminho.

Obrigado a Todos !!!

RESUMO

O crescimento da Internet exige continuamente investimentos direcionados à evolução da tecnologia TCP/IP. Estudos em sistemas de redes de computadores, comumente, utilizam a técnica da Simulação Digital. Nesse contexto, ambientes de simulação baseados em componentes despontam como alternativas que facilitam a construção e avaliação de desempenho de modelos de redes de computadores em geral. A abordagem de desenvolvimento orientado a componentes determina que uma aplicação seja constituída a partir de um conjunto de componentes interligados. Essa abordagem é uma evolução natural da abordagem de orientação a objetos, em que um componente corresponde a um conjunto de classes inter-relacionadas, com visibilidade externa limitada. Essa Dissertação de Mestrado apresenta uma especificação de componentes voltada para a modelagem de redes TCP/IP. Os componentes se apresentam genéricos de forma a representar as funcionalidades mínimas dos elementos básicos de uma rede TCP/IP, tais como fontes de tráfego, *hosts*, enlaces e roteadores. Essa especificação foi feita utilizando a Linguagem de Modelagem Unificada (UML) enfocando principalmente a fase planejar e elaborar, a fase de análise e a fase de projeto de alto nível (projeto arquitetural). As diversas fases do processo de desenvolvimento documentadas nesta Dissertação podem ser reutilizadas no desenvolvimento de ambientes de simulação orientados a componentes, como também podem ser reutilizadas no desenvolvimento de simuladores específicos, ambos voltados para a modelagem e avaliação de desempenho de redes de computadores com a tecnologia TCP/IP.

ABSTRACT

The growth of the Internet demand continuous investment directed to the evolution of the TCP/IP technology. Study of the systems of the computer networks, commonly, use the digital simulation technique. In this context, environments of simulation based in components emerge as alternatives that makes easy the build and performance evaluation of the general computer networks models. The approach to Component-Oriented software development determine that an application can be builded through a kit of components coupled. That approach to is a natural evolution of the approach to object- oriented, on that a component correspond to a inter-relacioned classes kit with limited extern view. That master dissertation presents a component specification concern to the modeling of the TCP/IP networks. The components are generics representing the minimum funcionalities of the TCP/IP network's basic elements, such as traffic sources, hosts, links and routers. That specification was made making use of Unified Modeling Language (UML) focalizing mainly the plan and elaborate phase, the analyze phase and design high-level phase (architecture design). Several phases of development process documented in this dissertation can be reusable in the development of the environment of the component-oriented simulator, as well, can be reusable in the development of the specific simulators, both concerns to the network performance modeling and evaluation with the TCP/IP technology.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

CAPÍTULO 1

1. Introdução

Este capítulo contextualiza esta Dissertação de Mestrado como um todo. Nele, apresenta-se, inicialmente, a motivação para a realização deste trabalho. Em seguida, apresentam-se os objetivos dessa dissertação e a sua relevância. Finalmente, apresentam-se a organização dos demais capítulos e os apêndices que compõe esta Dissertação.

1.1. Motivação

O crescimento contínuo da Internet obriga novos investimentos direcionados à tecnologia TCP/IP *(Transmisson Control Protocol/Internet Protocol*) para atender a demanda sempre crescente de serviços, voltados às aplicações mais complexas (multimídia, tempo real, etc.), exigidas por seus usuários [Tanenbaum, 97] [Comer,95].

Uma rede com tecnologia TCP/IP é caracterizada por uma armação topológica composta por *hosts* e roteadores interconectados por enlaces físicos que transportam unidades de informação. Os *hosts* comportam as aplicações dos usuários. Os roteadores são responsáveis pelo encaminhamento de informações (datagramas IP) entre redes interconectadas, viabilizando que uma informação gerada em uma rede possa chegar até *hosts* localizados em redes diferentes. Para isso, os roteadores devem ter conhecimento da topologia da rede e das rotas utilizadas para encaminhar as informações ou simplesmente ter um caminho padrão definido pelo gerente da rede.

A grande abrangência da Internet traz desafios para os projetistas de redes de computadores que buscam viabilizar sistemas de comunicação que otimizem a utilização da largura de banda dos canais de comunicação, utilizando eficientemente os meios de comunicação. Nesse sentido, destaca-se a comunicação *multicast*, na qual um usuário de uma rede de computadores envia informações para um grupo de usuários. Nesse tipo de comunicação o fluxo de informações gerado por uma determinada origem, atinge um ou mais *hosts* associados a um grupo de destino, fazendo com que não transitem pacotes idênticos pelo mesmo enlace e replicando-os somente quando necessário [Tanenbaum, 97] [Comer, 95].

A realização de estudos pertinentes as redes de computadores não é trivial, devido a gama de aspectos envolvidos e a simultaneidade em que eventos ocorrem nos diversos nós (*hosts* e roteadores) presentes em uma rede. Desta forma, é comum a busca por soluções através de técnicas de avaliação de desempenho.

Para viabilizar estudos de modelagem e de avaliação de desempenho direcionados aos Sistemas de Redes de Computadores, em particular, redes de computadores com a tecnologia TCP/IP, a técnica da Simulação Digital se faz geralmente presente [Kronbauer, 98]. Do ponto de vista prático, a Simulação Digital constitui-se no projeto e construção de modelos computadorizados de sistemas reais e propostos, visando compreender seus comportamentos em um conjunto de situações específicas [Kelton, 98].

A Engenharia de Software é uma sub área da Ciência da Computação que busca meios sistemáticos para a maximização da qualidade (características desejáveis) e da produtividade (agilidade) na atividade de desenvolvimento de software. Ela apoia o desenvolvimento de aplicativos em diversas outras áreas, tais como sistemas operacionais, redes de computadores e avaliação de desempenho (simulação digital) [Silva, 00].

A abordagem de desenvolvimento orientado a componentes determina que uma aplicação seja constituída a partir de um conjunto de componentes interligados. Essa abordagem é uma evolução natural da abordagem de orientação a objetos, em que um componente corresponde a um conjunto de classes inter-relacionadas, com visibilidade externa limitada [Silva, 00]. Ambientes de simulação baseados em componentes despontam como alternativas que facilitam a construção e avaliação de desempenho de sistemas em geral.

Um exemplo de um ambiente de simulação orientado a componentes é o Arena [Kelton, 98], que se apresenta atraente com recursos gráficos que viabilizam, com facilidade, a construção e a simulação de modelos que representam sistemas discretos em geral.

Uma definição geral de componente foi dada por [D´Souza, 98]: *componente é um pacote coerente de artefatos de software que pode ser desenvolvido independentemente e entregue como unidade e que pode ser composto, sem mudança, com outros componentes para construir algo maior*.

Usando essa definição, um componente pode incluir código executável, código fonte, projetos, especificações, testes, documentação, etc. Em termos de implementação, segundo [D'Souza, 98], *um componente é um pacote coerente de implementação de software que:*

- ✓ pode ser desenvolvido independentemente e entregue como unidade;
- ✓ tem interfaces explícitas e bem definidas para os serviços que oferece;
- ✓ tem interfaces explícitas e bem definidas para os serviços que requer;
- ✓ pode ser composto de outros componentes, talvez após a customização de algumas propriedades, mas sem modificar os componentes em si.

Construir ferramentas baseadas em componentes para a simulação de modelos de redes de computadores pode não ser uma tarefa simples. A experiência comprova que durante o processo de desenvolvimento normal de qualquer produto de software, as fases de análise (descrição do problema) e de projeto (descrição da solução) são as mais importantes e consomem a maior parte do tempo [Lula, 00] [Landin, 98]. Para reduzir efetivamente o tempo gasto durante o desenvolvimento de uma aplicação, é preciso fornecer não só reutilização de código, mas também a parte de **análise** e de **projeto**, de modo a reutilizar o conhecimento empregado e as principais decisões tomadas durante estas etapas [Freire, 00].

1.2. Objetivos

1.2.1. Objetivo Geral

Elaborar uma especificação de componentes voltados para a modelagem de redes TCP/IP. Os componentes propostos devem ser genéricos de forma a representar as funcionalidades mínimas dos elementos básicos de uma rede TCP/IP, tais como fontes de tráfego, *hosts*, enlaces e roteadores.

Essa especificação de componente poderá ser reutilizada em projetos de ambientes de simulação orientados a componentes voltados para a modelagem e avaliação de desempenho de redes de computadores, como também, poderá ser reutilizada em ambientes conhecidos como o Arena, permitindo tanto a construção de simuladores específicos, quanto a construção de novos componentes a serem integrados a estes ambientes.

1.2.2. Objetivos Específicos

- ✓ Realizar estudo sobre arquiteturas de redes de computadores, principalmente sobre a tecnologia TCP/IP;
- ✓ Estudar com detalhes as funcionalidades da camada internet da arquitetura TCP/IP, identificando seus elementos básicos, juntamente com os relacionamentos entre eles;

- ✓ Estudar a técnica da Simulação Digital e ferramentas de simulação voltadas para a avaliação de desempenho de redes de computadores;
- ✓ Estudar as técnicas utilizadas pela engenharia de software para desenvolvimento de componentes de software;
- ✓ Propor uma especificação de componentes de software genéricos, que representem as funcionalidades mínimas dos elementos de uma rede TCP/IP. Essa especificação deve ser feita utilizando a Linguagem de Modelagem Unificada (UML) [Fowler,97] [Larman,98], seguindo um processo de desenvolvimento apresentado em [Larman,98], enfocando a fase planejar e elaborar, a fase de análise e a fase de projeto de alto nível (projeto arquitetural).

1.3. Relevância

Essa Dissertação apresenta uma especificação de componentes que representam elementos da tecnologia TCP/IP em uma linguagem de modelagem bastante difundida (UML). Essa especificação possibilita que profissionais de Informática, principalmente analistas de modelagem e desenvolvedores de softwares de simulação para redes de computadores, possam reutilizá-las em seus projetos específicos. Observa-se que, embora a tecnologia TCP/IP seja muito conhecida e difundida, não foi encontrado na literatura especializada consultada, estudos direcionados à modelagem dessa tecnologia usando a abordagem Orientada a objetos.

Existem ambientes de simulação, como por exemplo o Arena [Kelton,98] e o SAVAD [Cabral,93] (utilizados em estudos de avaliação de desempenho do Grupo de Redes de Computadores da UFPB), que são orientados a componentes. Embora esses ambientes sejam passíveis à agregação de novas funcionalidades, eles não detém mecanismos explícitos para a modelagem de redes TCP/IP. Nesse contexto, a documentação gerada por esta dissertação serve como ponto de partida para implementação das funcionalidades básicas da tecnologia TCP/IP em tais ambientes de simulação ou mesmo em outros ambientes em desenvolvimento que sejam orientados a componentes. Esta documentação busca intercalar notações informais, semi-formais e formais, sob forma de narrativas, figuras, diagramas, tabelas, etc., o que facilita a compreensão tanto da especificação em si quanto do funcionamento da camada internet. Haja visto que as *Request For Comments* (RFC) da publicadas pela *Internet Engineering Task Force* (IETF) tem sido publicadas em modo texto, sem utilização de uma linguagem semi-formal ou formal, o que dificulta a compreensão dos protocolos que estão sendo especificados.

O fato da abstração dos elementos de redes TCP/IP ter sido baseada no paradigma orientação a objetos, propicia a utilização de uma abordagem voltada ao reuso, possibilitando que se alcance maior produtividade, qualidade e economia do sistema (software), a ser desenvolvido: produtividade, porque será minimizada uma das etapas mais dispendiosas do processo de desenvolvimento, a especificação do sistema; qualidade, porque partindo de uma especificação existente pode-se realizar um número maior de refinamentos no sistema; e economia, devido à

minimização do tempo a ser despendido. Este paradigma também permite que novas funcionalidades sejam agregadas com facilidades ao sistema em desenvolvimento .

Finalmente, há uma interseção de três áreas de pesquisa envolvidas neste estudo: Redes de Computadores, Modelagem e Avaliação de Desempenho e Engenharia de Software. Isto caracteriza a multidisciplinaridade, aspecto importante em um trabalho a nível de mestrado.

1.4. Organização da Dissertação

Esta dissertação está organizada em 05 capítulos e 04 apêndices.

Seguem, de forma resumida, os conteúdos dos demais capítulos e apêndices desta Dissertação:

O capítulo 2 apresenta a técnica da Simulação Digital, mostrando como esta se aplica em estudos de modelagem e de avaliação de desempenho de sistemas discretos, nos quais redes de computadores se inserem.

O capítulo 3 apresenta um estudo da camada internet da tecnologia TCP/IP.

O capítulo 4 apresenta a especificação de componentes que representam elementos básicos de uma rede TCP/IP. Essa especificação foi feita utilizando a Linguagem de Modelagem Unificada (UML), seguindo um processo de desenvolvimento apresentado em [Larman,98], enfocando a fase planejar e elaborar, a fase de análise e a fase de projeto de alto nível (projeto arquitetural).

O capítulo 5 apresenta considerações finais sobre o trabalho desenvolvido nesta Dissertação, como também sugestões de continuidade desta.

O Apêndice A constitui-se de um estudo prévio ao capítulo 3, apresentando um estudo sobre arquitetura de redes de computadores voltando para a tecnologia TCP/IP.

Os Apêndice B e C constituem-se de estudos complementares ao capítulo 3, apresentando, respectivamente, exemplos de protocolos e algoritmos de roteamento.

Finalmente, o Apêndice D apresenta os acrônimos usados nesta Dissertação.

Capítulo 2

2. Simulação Digital - uma abordagem

Este capítulo introduz a técnica da Simulação Digital e a sua aplicabilidade em estudos de avaliação de desempenho de sistemas discretos, nos quais redes de computadores se inserem.

A seção 2.1 apresenta as formas de se avaliar o desempenho de um sistema. A seção 2.2 introduz a técnica da Simulação Digital. A seção 2.3 apresenta uma metodologia adotada em estudos de simulação. Uma introdução à simulação orientada a objetos é apresentada pela seção 2.4. Finalmente, a seção 2.5 apresenta exemplos de ambientes de simulação de alto nível, referências importantes no escopo desta Dissertação.

2.1. Introdução

Quando se está envolvido com um sistema, independentemente de sua natureza, é imprescindível verificar o seu comportamento de acordo com as mais diversas situações em que ele pode se encontrar, viabilizando a tomada de decisões apropriadas, tais como sua desativação ou otimização.

Para analisar o desempenho de sistemas discretos, tais como redes de computadores, é preciso definir um conjunto de medidas de desempenho relevantes de interesse, observando o comportamento destas medidas face às variações na demanda da sua utilização. Esta demanda é

caracterizada pelo tipo de aplicação na entrada do sistema (tráfego) [Celestino, 90]. A análise de desempenho desses sistemas pode ser realizada de duas formas:

- ✓ através de campanhas de medição (*benchmarks*) em sistemas já existentes;
- ✓ a nível de projeto, por intermédio de um modelo do sistema proposto ou existente.

A segunda alternativa é muito interessante, devido à possibilidade de se fazer projeções de diversas configurações e situações do sistema a um baixo custo. Com isso, minimiza-se as probabilidades de erros de dimensionamento e utilização do sistema [Souto, 93]. Um modelo é criado através da abstração simplificada de um sistema, contudo este deve captar características fundamentais do comportamento deste sistema. Esta alternativa tipicamente é viabilizada a partir de duas alternativas: Simulação Digital e Estudos Analíticos (Teoria dos Processos Estocásticos / Teoria das Filas) [Damoun, 79].

A solução analítica é mais econômica e eficiente, entretanto muitas vezes sua aplicação é limitada pela complexidade do sistema que está sendo modelado. Para este tipo de situação, em certas ocasiões, a única solução possível é a Simulação Digital [Dias, 92].

2.2. Simulação Digital

A Simulação Digital é uma técnica usada para prever o comportamento de sistemas reais ou hipotéticos. Ela utiliza um modelo para obter relações entre as medidas de desempenho de interesse, permitindo a experimentação automatizada com sistemas que não foram concebidos ou de difícil manuseio. Ela também permite a experimentação repetida de um sistema, sob condições controladas, para otimizar seu desempenho [Dias, 92].

Um sistema é um conjunto de partes organizadas funcionalmente para formar um todo [Sauvé, 86]. Neste sentido, uma parte do sistema que pode ser tratada como um sistema isolado é chamado de subsistema. Um sistema é portanto formado por um conjunto de subsistemas [Almeida, 99]. Por exemplo, sendo a Internet um sistema, um roteador nela presente, pode ser visto como um subsistema. Sob outro ponto de vista, um roteador pode ser visto como um sistema e os protocolos, que o compõe, seus subsistemas.

Um modelo é um conjunto de aproximações e suposições, ambas estruturais e quantitativas, sobre as funcionalidades de um sistema. De uma maneira mais simples, pode-se afirmar também que um modelo é uma abstração de um sistema, ou seja, é a observação das características relevantes de acordo com o problema a ser resolvido e do ponto de vista de quem está modelando [Almeida, 99].

Na elaboração de um modelo, é necessário decidir qual o nível de detalhamento desejado. Quanto mais abstrato menos detalhado será o modelo. Desta forma, é interessante que se defina vários níveis de abstração, pelos quais um sistema pode ser representado. O nível mais alto apresenta uma visão geral do sistema e o nível mais baixo contempla o modelo com maiores detalhes das funcionalidades do sistema [D'Souza, 98]. A partir do nível mais alto, através de

vários níveis, chega-se ao nível mais baixo pela realização de refinamentos no modelo. Desta forma, torna-se óbvio que quanto mais baixo o nível de abstração, maior o tempo despendido para a elaboração do modelo.

Em simulação, geralmente, a construção da lógica de um modelo despende de 30% a 40% do tempo total do projeto de simulação. No início esta porcentagem é mais elevada devido à falta de experiência das pessoas envolvidas, tanto no manuseio da ferramenta a ser utilizada, quanto na coleta dos dados necessários. Por isso é recomendável que as pessoas envolvidas nos projetos tenham pelo menos um bom conhecimento do processo a ser modelado. Ao realizar a simulação de um sistema, a qualidade do modelo a ser construído e os dados a serem aplicados a este modelo irão refletir nos resultados obtidos. Bons modelos e dados coerentes levam a resultados satisfatórios, e vice-versa [Kelton, 98].

Em seus primórdios, a simulação digital era extremamente complicada, devido à necessidade de modelagem matemática dos sistemas e a implementação de algoritmos em linguagens de programação de propósito geral (ex. FORTRAN). O grau de dificuldade diminuiu com o surgimento das linguagens específicas para simulação, tais como, GPSS, SIMSCRIPT, SLAM, e SIMAN [Kelton, 98]. Entretanto, a maturidade na área de simulação está sendo atingida com os simuladores de alto nível. Ferramentas de simulação geralmente trazem um conjunto de componentes pré codificados, com a habilidade para personalizar ou adicionar novos módulos. Estas ferramentas tipicamente oferecem interface gráfica com o usuário, menus e diálogos, todos operáveis através de *clicks* intuitivos no mouse, e permitem a criação e animação de modelos de sistemas de forma flexíveis. Existem vários exemplos de ferramentas de simulação de domínio público (ex. SAVAD [Cabral, 92], ns [ns, 98], MARS [Alaettinoglu, 98]) e comerciais (Arena [Kelton, 98] e BONeS [Cadence, 98].

2.3. Metodologia para Estudos de Simulação

Em um estudo de simulação deve-se definir adequadamente a metodologia a ser utilizada na sua realização. Esta seção apresenta uma seqüência de atividades de um estudo de simulação (figura 2.1) e considerações sobre o relacionamento entre estas. Tais atividades foram definidas em [Kelton, 98] e em [Lobão, 96]. É importante ressaltar que as fontes bibliográficas citadas voltam seus exemplos e considerações para a criação de modelos de simulação das área de Engenharia da Produção, Logística, entre outras. Nesta seção, contudo, foram inseridas algumas considerações sobre o estudo de simulação em redes de computadores.

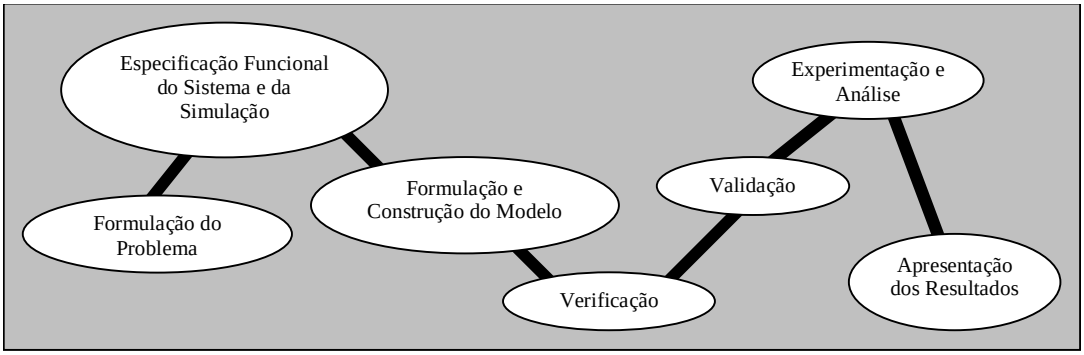


Figura 2.1: Atividades do Estudo de Simulação.

A relação entre as atividades a serem realizadas estão diretamente associadas à imensa gama de particularidades de cada sistema a ser avaliado. Portanto, o estudo de simulação pode ser realizado como se fosse movimentos em uma espiral, aproximando-se e afastando-se de seu centro de acordo com a clareza de seus objetivos [Kelton, 98]. As atividades encontram-se em pontos definidos nos anéis da espiral e o centro pode ser abstraído como o sucesso alcançado. Em outras palavras, pode-se dizer que o estudo de simulação deve ser realizado em vários ciclos (anéis da espiral) de desenvolvimento interativos, onde pode ser observado o aumento do nível de detalhamento a cada ciclo realizado (refinamento).

De acordo com a figura 2.1, as atividades básicas para avaliação de desempenho de redes de computadores, utilizando-se da simulação, são as seguintes [Kelton, 98]: Formulação do Problema; Especificação Funcional do Sistema e da Simulação; Formulação e Construção do Modelo; Verificação; Validação; Experimentação e Análise; Apresentação e Preservação dos Resultados.

2.3.1. Formulação do Problema

Na resolução de problemas, o primeiro passo é definir e formular o problema. No mundo real, raramente os problemas a serem resolvidos estão completamente claros ou definidos no papel. Nestes casos, não se pode dizer que a simulação é a ferramenta correta para a solução [Kelton, 98]. Estudos bibliográficos e diálogos com pessoas interessadas na solução (clientes) são fundamentais para se formular claramente o problema. Uma classificação interessante que interferirá na estratégia de definir e atacar o problema foi descrito na figura abaixo:

Tipos de Problemas	Informações Relevantes
Avaliação de sistemas que não correspondem às expectativas do usuário (ex. rede local sobrecarregada de serviços e usuários);	Dados Reais
Avaliação de especificações de sistemas que não foram implementados, ou não se tem acesso aos dados das instâncias atuantes (ex. análise comparativa ou validativa de protocolos de	Dados estimados
Avaliação dos limites de sistemas para conhecer todas as limitações e potencialidades do mesmo (ex. gerência pró-ativa de redes de computadores).	Dados Reais e Dados estimados

Figura 2.2: Informações utilizadas no estudo de Simulação.

Após identificar o sistema em uma das classes de problemas, mostradas na figura 2.2, deve-se estabelecer os limites iniciais do mesmo. No contexto das redes de computadores o problema do sistema pode ser delineado a nível de processos, protocolos, serviços, aplicações, tecnologias de rede, etc.. Entretanto, é importante observar o sistema de acordo com as funcionalidades definidas nas camadas da arquitetura de redes em questão(ex.: TCP/IP, RM-OSI).

Os limites do sistema são aprimorados à medida que os conhecimentos sobre o sistema e seu problema são aprofundados. Portanto esta tarefa deve ser delineada com atenção, pois a expansão ou contração desses limites influenciarão diretamente nos procedimentos de resolução do mesmo [Kelton, 98].

A partir da definição dos limites iniciais, parte-se para a próxima etapa da formulação do problema: definir métricas de desempenho. Pode-se considerar duas classes de métricas de desempenho, nas quais deve-se centrar [Kelton, 98]:

- ✓ Métricas usadas para medir a *Qualidade do Sistema* sobre o estudo;
- ✓ Métricas usadas para medir o *Sucesso do Estudo*.

Depois de estabelecer como o desempenho do sistema será medido, deve-se descobrir a existência de valores *"Baseline"* atuais para estas métricas. Caso seja impossível (ex.: sistema novo), busca-se sistemas existentes similares para se fazer estimativas (ver figura 2.2).

Enfim, tendo em mãos as medidas (métricas) atuais deve-se descobrir "Quais as expectativas do cliente?", ou seja, traçar os objetivos do estudo. Os objetivos estando definidos, pode-se partir para a próxima atividade (Especificação Funcional do Sistema e de Simulação). É interessante frisar que mesmo os objetivos estando definidos, estes podem ser alterados posteriormente, caso novas informações relevantes sejam descobertas na execução da próxima atividade [Kelton, 98].

2.3.2. Especificação Funcional do Sistema e da Simulação

Embora a especificação funcional não seja de interesse direto desta Dissertação, a sua menção nesta seção deve-se a esta especificação ter sido tradicionalmente usada em estudos de simulação.

Para iniciar esta etapa (atividade), o problema já deve ter sido formulado e os objetivos do estudo definidos, mesmo que estes não sejam considerados definitivos [Lobão, 96].

Nesta etapa, busca-se entender o funcionamento do sistema detalhadamente e definir os parâmetros pertinentes à abstração deste sistema para a simulação (ênfase aos dados de entrada e saída da simulação). O resultado desta etapa pode ser expresso em um documento chamado Especificação Funcional.

O processo de desenvolvimento de uma especificação pode ter muitas formas dependentes de fatores como [Kelton, 98]:

- ✓ Abrangência do estudo;
- ✓ Relacionamento das partes envolvidas (analista e cliente);
- ✓ Habilidade de ambas as partes em chegar a um consenso dos detalhes deste primeiro estágio.

Entretanto, se um indivíduo fizer ambos os papéis (analista e usuário), esta etapa pode ser combinada com a próxima (Formulação e Construção do Modelo). Embora seja necessário definir e entender o sistema completamente, o desenvolvimento de uma especificação formal da simulação não é obrigatório. A obrigatoriedade deste documento é justificada quando o analista é um consultor externo à empresa (ou sistema) do usuário, pois ele implicaria na comunicação formal entre as partes. A especificação é importante quando mais de uma pessoa trocar informações sobre o assunto [Kelton, 98].

2.3.3. Formulação e Construção do Modelo

A construção do modelo se restringe a três partes [Kelton, 98]:

- ✓ **Lógica de Controle:** nesta etapa começa a construção da lógica propriamente dita do software. Aconselha-se a utilização de técnicas de programação, tais como algoritmos ou fluxogramas para expressar a idéia da lógica que se deseja construir;
- ✓ **Dados Necessários:** a especificação dos dados a serem utilizados no modelo deve ser detalhada;
- ✓ **Flexibilidade e Nível de Detalhamento:** antes da construção do modelo deve-se definir o nível de detalhamento do mesmo. Quanto mais detalhado, mais complexo e portanto, menos flexível.

2.3.4. Verificação

Uma vez que o modelo esteja construído, parte-se para verificar se a lógica implementada está agindo de acordo com o previsto no projeto. Caso contrário, deve-se fazer os ajustes necessários para tal. O processo de verificação consiste em isolar e corrigir erros não intencionais no modelo. A complexidade do processo de verificação depende do tamanho do modelo, e a parte mais difícil não é corrigir erros, e sim isolá-los.

2.3.5. Validação

É uma etapa muito parecida com a anterior, mas que nunca deve precedê-la, no máximo ser realizada simultaneamente. Ela consiste em certificar se o sistema modelado está em sintonia com o sistema real, quanto ao comportamento e quanto aos resultados obtidos. Se os resultados destoam muito do real, provavelmente os dados de entrada podem estar com problemas ou até mesmo alguma consideração lógica do modelo.

O envolvimento dos componentes, tais como: entidades, recursos, lógica de controle, estatísticas, uso de animação e dados em um modelo racional, influenciam nos processos de verificação e validação.

2.3.6. Análise

Considerando que o modelo esteja se comportando como o sistema real, inicia-se a busca dos objetivos do objeto de estudo da avaliação de desempenho. Aqui, identifica-se problemas, seu grau de importância, e viabilidades para a resolução dos mesmos.

2.3.7. Apresentação e Preservação dos Resultados

Assim como todo projeto, deve haver uma apresentação às pessoas interessadas na área em questão. Algumas considerações quanto à apresentação [Kelton, 98]:

- ✓ Ser breve e simples para um entendimento fácil;
- ✓ Enfocar a apresentação dos resultados significativos alcançados;
- ✓ Direcionar a linguagem de apresentação à audiência, lembrando-se que muitos deles não participaram do projeto, e
- ✓ Apresentar razões para os resultados expostos, senão a apresentação perderá o seu embasamento.

2.4. Simulação Orientada a Objetos

Sob um ponto de vista superficial, a expressão “Orientação a Objetos” significa que o software é organizado como uma coleção de objetos separados que incorporam, tanto a estrutura,

quanto o comportamento dos dados. As principais características do paradigma orientação a objetos, geralmente, incorporam quatro aspectos [Rumbaugh, 94]:

- ✓ **Identidade**: significa que os dados são subdivididos em entidades discretas e distintas, denominadas objetos;
- ✓ **Classificação**: significa que os objetos (instâncias da classe) com a mesma estrutura de dados (atributos) e o mesmo comportamento (operações) são agrupados em uma classe. Uma classe é uma abstração que descreve propriedades importantes para uma aplicação ignorando o restante. Cada classe descreve um conjunto possivelmente infinito de objetos individuais;
- ✓ **Polimorfismo**: significa que a mesma operação pode atuar de modos diversos em classes diferentes. Uma operação é uma ação ou transformação que um objeto executa ou a que ele está sujeito. Uma implementação específica de uma operação por uma determinada classe é chamada de método. Cada classe tem seus métodos específicos para implementar uma determinada operação;
- ✓ **Herança**: é o compartilhamento de atributos e operações entre classes, com base em um relacionamento hierárquico. Uma classe pode ser definida de forma abrangente e depois refinada em sucessivas subclasses mais definidas. Cada subclasse incorpora, ou herda, todas as propriedades de sua super classe e acrescenta suas próprias e exclusivas características.

Desde a metade da década passada, tem havido um sensível crescimento na pesquisa, desenvolvimento e aplicação de ferramentas de simulação orientada a objetos (*Object Oriented Simulation* – OOS) [Roberts, 94].

Sistemas complexos são, por natureza, maiores e mais heterogêneos que outros sistemas, além do que, freqüentemente, os modelos são desenvolvidos e mantidos por várias pessoas. Portanto, conforme a complexidade do sistema aumenta, aumentam também os benefícios da OOS [Almeida, 99].

A vantagem mais tipicamente citada para se usar esta abordagem é a capacidade de modelar sistemas usando entidades que são naturais ao sistema físico. Desta forma, por exemplo, um sistema de trânsito, que consiste de pedestres, automóveis e semáforos, pode ser diretamente representado em seu modelo por essas mesmas entidades. Isto possibilita ao analista de modelagem uma maior simplicidade e flexibilidade durante a concepção de seu modelo, permitindo, inclusive, decompor o sistema em subsistemas através da herança de atributos e de comportamento e daí interconectá-los a diferentes subsistemas a fim de formar novos sistemas [Almeida, 99].

Além da modelagem mais natural e, portanto mais simples, a OOS também se propõe ao desenvolvimento mais rápido de modelos através da utilização de modelos pré-definidos em bibliotecas. Neste caso, basta ao analista de modelagem utilizar ou adicionar novos atributos ou funcionalidades a determinados elementos que compõem a biblioteca. Também é possível ao

analista de modelagem criar aqueles elementos que não existem na biblioteca e, eventualmente adicioná-los [Almeida, 99].

Um avanço da engenharia de software está relacionado ao desenvolvimento de software orientado a componentes. Um componente pode ser interpretado como um conjunto de artefatos coerentes de conceitos (especificados de forma independente) que são tratados como uma unidade (com interfaces bem definidas) e podem ser utilizados juntamente com outros componentes para se construir algo maior [D'Souza, 98].

Existem algumas diferenças com relação à construção de projetos baseados em componentes de projetos baseados em objetos. Quando se constrói um projeto de componentes não é necessário saber como eles são representados (como objetos, como instâncias de uma classe) nem se preocupar como os conectores entre componentes trabalham. Assim, como em programas orientados a objetos, cada componente é uma coleção de software. Também como em programas orientados a objetos, onde objetos devem acessar informações guardadas por outros objetos, componentes se intercomunicam através de interfaces bem definidas para preservar encapsulamento mútuo. De fato, componentes podem ser implementados como objetos em uma linguagem orientada a objetos. Desta forma, as diferenças entre projeto baseado em componentes e projeto baseado em objetos são principalmente em grau e escala e não intrínsecas para cada tipo de projeto [D'Souza, 98].

- ✓ Componentes freqüentemente usam armazenamento persistente; objetos tipicamente trabalham somente na memória principal, e a persistência é tratada separadamente;
- ✓ Componentes tem uma rica faixa de mecanismos de intercomunicação, tal como eventos e "*workflows*", possibilitando fácil composição entre as partes. Objetos tem somente as mensagens básicas da orientação a objeto;
- ✓ Componentes oferecem granularidade maior que os objetos e podem ser implementados como múltiplos objetos de diferentes classes. Geralmente eles permitem ações complexas em suas interfaces, em vez de simples mensagens;
- ✓ Um Pacote de componentes, por definição, inclui definições de interfaces que ele provê e interfaces que ele requer. A definição clássica de classes foca as atenções apenas nas interfaces providas.

Componentes, assim como objetos, interagem através de interfaces polimórficas. Todas as técnicas de modelagem se aplicam igualmente em ambos os casos. Também pode-se falar sobre "instâncias de componentes", "tipos de componentes" e "classes de componentes" [D'Souza, 98].

A reusabilidade de software é reconhecida como uma importante maneira de aumentar a produtividade no processo de desenvolvimento de software. Para maximizar a reusabilidade, é preciso fornecer não só reutilização de código, mas também de análise e de projeto, de modo a

reutilizar o conhecimento empregado e as principais decisões tomadas durante estas etapas [Freire, 00].

No contexto da reutilização de componentes, existem situações em que pode-se modificar e/ou estender componentes para que estes atendam as características específicas de uma dada aplicação. A tarefa de alteração de um componente pode se tornar mais dinâmica se o mesmo tiver sido desenvolvido com para ser de alterado. Este é o caso do "*Framework* de componentes", o qual é um componente que prevê o acoplamento de outros componentes e que é modificado a partir da troca dos componentes a ele acoplados [Silva, 00].

Para o desenvolvimento de um projeto de software deve-se utilizar alguma metodologia de desenvolvimento (processo de desenvolvimento mais linguagem de modelagem). Um processo e desenvolvimento define **quem** faz **o que**, **quando** e **como**, para atingir um certo objetivo. Uma visão macro de um processo de desenvolvimento apresentado em [Larman, 98] consiste de 3 grandes fases:

- ✓ **Planejamento e Elaboração**: esta fase consiste basicamente nas seguintes atividades:
  - Elaborar relatório inicial de investigação (Domínio do Problema);
  - fazer levantamento de requisitos funcionais e não funcionais;
  - ilustrar *use-cases*;
  - definir o modelo conceitual inicial;
  - apresentar um protótipo para ajudar a compreensão do problema;
  - elaborar um glossário de termos.
- ✓ **Construção**: essa fase envolve repetidos ciclos de desenvolvimento. Os passos principais referem-se à análise e ao projeto;
- ✓ **Implantação**: refere-se à implantação do sistema para uso.

A *Unified Modeling Linguage* (UML) é uma linguagem gráfica para visualização, especificação, construção e documentação de artefatos de sistemas de software. A UML oferece uma forma padrão para escrever projetos de sistemas, incluindo coisas conceituais, tal como, funções de processos do negócio, como também, coisas concretas como expressões de linguagens de programação, esquemas de banco de dados e componentes de software reutilizáveis. Ela está sendo indicada na realização da análise e projeto de softwares, pois é o resultado de anos de trabalho no sentido de unificação dos métodos mais utilizados para documentação de desenvolvimento de software [OMG, 00]. A UML foi criada por James Rumbaugh, Ivan jacobson e Grady Booch, entretanto hoje está sob responsabilidade da *Object Modeling Group* (OMG). A UML é composta por vários diagramas que facilitam a interação entre o analistas do software (modelo), os desenvolvedores e os usuários. Os diagramas utilizados neste trabalho são apresentados no capítulo 4.

2.5. Ambientes de Simulação

Esta sessão apresenta dois ambientes de simulação, os quais são referências importantes nesta Dissertação. O primeiro, denominado SAVAD, desenvolvido pelo Grupo de Redes de Computadores da Universidade Federal da Paraíba (UFPB) [Cabral,92] e o Arena [Kelton,98], um ambiente de simulação de alto nível, produto adquirido pela UFPB.

2.5.1. Ambiente SAVAD

O SAVAD (Sistema Especialista para Avaliação de Desempenho de Modelos de Redes de Filas) é um ambiente de modelagem integrado, onde partindo de um modelo de redes de filas proposto pelo usuário, escolhe e aplica o método mais adequado para solucionar este modelo, fornecendo as medidas de desempenho relevantes. Para isso, o SAVAD utiliza técnicas analíticas, baseadas na Teoria das Filas e a técnica da Simulação Digital [Souto, 93] [Oliveira, 95].

O SAVAD foi desenvolvido usando as linguagens de programação C++ [Zortech, 90] e Prolog [Clocksin, 81] em ambiente compatível com IBM-PC. As características pelas quais foi projetado exploram diversos aspectos da abordagem orientada a objetos, facilitando, com isso, o desenvolvimento de versões futuras [Oliveira, 95].

Para a modelagem de sistemas de redes de filas são utilizados elementos que descrevem o comportamento do problema a ser analisado. O SAVAD apresenta vários elementos para permitir a modelagem de sistemas de redes de filas de forma flexível. Os elementos são [Cabral, 92]:

- ✓ Clientes: são entidades que circulam através dos elementos do modelo, solicitando serviços;
- ✓ Estações de Serviço: representam os recursos de um modelo de redes de filas (ex.: canais de transmissão em uma rede de computadores);
- ✓ Fontes: geram os clientes em modelos de redes abertas;
- ✓ Sorvedouros: eliminam clientes em modelos de redes abertas;
- ✓ Pontos de Controle: permitem controlar o fluxo de clientes nos modelos de redes de filas;
- ✓ Classes: são usadas para determinar tipos diferentes de clientes em circulação no modelo, e
- ✓ Rotas: estabelecem os diversos caminhos por onde os clientes trafegam. O modelo de redes de filas é definido pelo estabelecimento de rotas com classes especificadas, unindo os diversos elementos**.** O roteamento no SAVAD é fixo.

O SAVAD é um ambiente de modelagem integrado para solucionar modelos que exibem contenção de recursos, particularmente modelos que representam redes de computadores.

Durante o processo de construção do modelo, o SAVAD limita-se ao definir rotas fixas para o tráfego de informações.

2.5.2. O Ambiente Arena

O ambiente de simulação de propósito geral ARENA foi desenvolvido, em 1993 para a plataforma Microsoft Windows, pela empresa Norte Americana *Systems Modeling Corporation* [Takus, 94], sendo representada no Brasil pela empresa *Paragon Software*. Dentre as características do simulador, destaca-se sua integração com outras ferramentas que fazem uso da tecnologia *ActiveX*, o que garante sua conectividade com outros aplicativos tipo Excel, Word, etc..

Tendo sido desenvolvido na linguagem Visual C++, utilizando técnicas de orientação a objetos e recursos da *Microsoft Foundation Classes* (MFC), o Arena faz uso das facilidades oferecidas pela interface gráfica do ambiente (barras de ferramentas, caixas de diálogo, etc..) [Wagner, 00].

O Arena também gera código da simulação na linguagem SIMAN [Pegdrm, 95], linguagem específica para simulação, o que permite a depuração de erros da lógica do modelo com facilidade [Takus 97]. Ainda, o Arena combina a facilidade de uso encontrada nos simuladores de alto nível com a flexibilidade da linguagem de simulação SIMAN e permite a utilização de códigos de linguagens de propósito geral, tais como *Visual Basic*, FORTRAN e C/C++[Wagner, 00].

O Arena apresenta uma estrutura hierárquica baseada em *templates*[1], que englobam painéis, que por sua vez englobam componentes. Os componentes de painéis, pertencentes a um mesmo *template* ou a *templates* distintos, podem ser combinados entre si, permitindo construir uma variedade maior de modelos [Kelton, 98].

Componentes são elementos de software auto-suficientes e reutilizáveis, que podem interagir entre si, seguindo um grupo de regras preestabelecidas [Englander, 97]. Estes podem ser utilizados tanto na construção de aplicações, como também na construção de outros componentes. Desta forma, o Arena apresenta um grande número de alternativas verticais (figura 2.3) para construção de modelos, possibilitando ao usuário trabalhar com componentes específicos de alto nível, juntamente com a elaboração de rotinas em linguagens de propósito geral. Por um lado, o usuário tem acesso a todos os recursos de simulação, tais como, suporte à decisão, animação gráfica e lógica de modelagem sem ter que desenvolver uma única linha de código. Um usuário experiente pode aperfeiçoar sua produtividade e compartilhar seu conhecimento com outros, ao construir componentes reutilizáveis [SMC, 96].

[1] Template é um módulo composto por um conjunto de painéis que agrupam componentes (elementos).

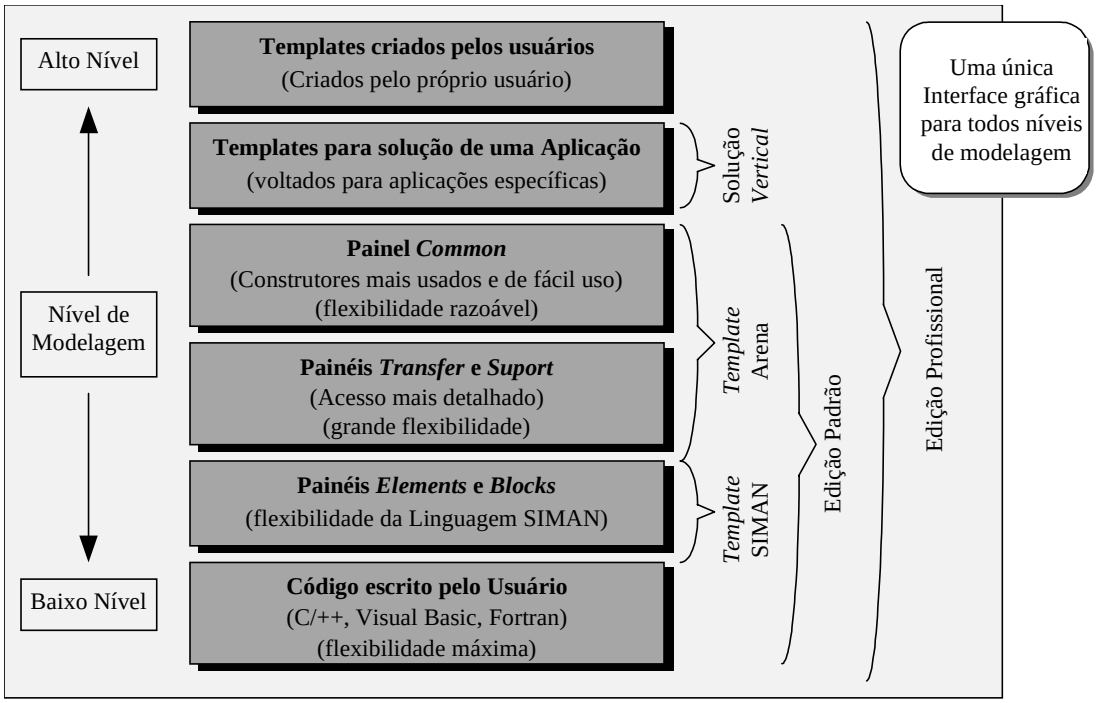


Figura 2.3: Estrutura Hierárquica do Arena [Kelton, 98].

Atualmente, o Arena se encontra disponível em duas versões: Profissional e Acadêmica, sendo que esta última é distribuída de forma gratuita. As diferença entre as versões são as limitações impostas na versão Acadêmica, que limita a utilização de modelos com no máximo de 150 elementos, além de não permitir a criação de novos *templates* (conjunto de componentes). A versão profissional apresenta dois ambientes de trabalho, descritos a seguir.

O Ambiente *Arena Professional Edition* é designado à construção, pelo próprio usuário, de novos *templates* específicos para as aplicações do usuário. Por exemplo, um *template* voltado para a avaliação de desempenho de redes de computadores pode ser construido e anexado ao arena. Este *template*, por exemplo, poderia ter um painel composto por elementos (componentes) que abstraem particularidades da tecnologia ATM, outro painel com funcionalidades de protocolos da arquitetura TCP/IP, etc..

O Ambiente *"Arena Standard Edition* é designado para a construção, simulação e análise dos modelos do usuário. Ele é composto das seguintes ferramentas [Kelton, 98] [Wagner, 00]:

- ✓ **Simulador:** o ambiente em si, com interface padrão Windows, é destinado à construção, execução e manipulação de modelos. Possui, entre outros, elementos de animação, modelagem e interação com outros aplicativos relevantes;
- ✓ **Analisador de Entradas (*Input Analyzer*):** Dado um arquivo do tipo texto, contendo os intervalos de perturbações ao sistema e/ou suas durações, este programa descreve seu comportamento através de equações matemáticas. O arquivo, tanto poderá ser gerado através do registro de eventos do dispositivo (*log*), como manualmente através de editor de texto. Outra função do aplicativo é a

geração de um arquivo do tipo texto baseado em distribuições estatísticas com o intuito de garantir a utilização de valores iguais em simulações diferentes. As distribuições estatísticas utilizadas pelo programa são: *Beta, Empirical, Erlang, Exponencial, Gama, Johnson, Lognormal, Normal, Poisson, Triangular, Uniforme* e *Weibull* [Kelton, 98].

- ✓ **Analisador de Saídas (*Output Analyzer*):** Responsável pela interpretação dos resultados da simulação que são coletados através do componente '*Statistics*' do painel *common*. Este programa permite a geração de tabelas, gráficos de barra ou pontos, histogramas, determinação de intervalos de confiança das amostras apresentadas e exportação dos valores para arquivos texto padrão, para serem analisados por outros aplicativos.
- ✓ **Gerenciador de Cenários (*Scenario Manager*):** tem como objetivo a execução em lote de um modelo, podendo o modelo ser exposto às diferentes condições. Os resultados tanto podem ser analisados através da ferramenta *Output Analyzer,* como através de relatórios gerados pelo próprio programa.

Buscando proporcionar modelos que se aproximem do comportamento do sistema que esta sendo modelado, o Arena dispõe de mecanismos e artifícios que permitem a integração com outros ambientes (Java, Delphi), conciliando as facilidades disponíveis em simuladores de alto nível, com a flexibilidade proposta pelas linguagens de programação, conforme apresentado a seguir [Wagner, 00]:

- ✓ **Fonte de dados externa**: Permite ao ARENA utilizar registros de eventos criados pelo usuário, ou coletados de dispositivos presentes no sistema real, ou até mesmo por outra aplicação. Este recurso permite submeter o modelo a situações reais. Tal integração pode ocorrer utilizando elementos próprios para leitura e escrita em arquivos textos, como os componentes *READ* e *WRITE* localizados no painel *Support* do *template* ARENA, ou estruturas de acesso à base de dados fornecidas pelo ambiente operacional, tipo *Data Access Objects* (DAO). Esta estrutura permite a utilização e recuperação de dados localizados em bancos de dados como *SQL Server, Oracle, Informix*, ou em documentos do *Microsoft Office*;
- ✓ **ActiveX**: Esta tecnologia tem como objetivo automatizar ações que deveriam ser realizadas pelo operador. Através desta, o modelo é capaz de responder aos eventos do sistema operacional, tais como abertura e fechamento de documentos, início e término de simulação, entre outros, permitindo inclusive a integração com outros aplicativos projetados com este fim [Kelton, 98]. Este recurso utiliza como linguagem nativa de programação o *Visual Basic for Aplications* (VBA), fazendo com que qualquer objeto (ex. aplicativos do *Microsoft Office*) devidamente registrado no sistema possa ser acessado pelo simulador;

Código Externo: Além dos recursos apresentados anteriormente, o Arena permite o acoplamento de rotinas desenvolvidas em C++ ou FORTRAN. Esta flexibilidade auxilia o projetista

no desenvolvimento de tarefas complexas que não possam (ou não seja interessante) ser representadas utilizando os recursos de alto nível ou da linguagem SIMAN (ex. funções recursivas). Para isso, o Arena é acompanhado por um conjunto de bibliotecas para os seguintes compiladores Microsoft Visual C++ 4.0; Microsoft PowerStation 32 FORTRAN 4.0; Watcom 10.5 C++; Watcom 10.5 FORTRAN

Capítulo 3

3. Camada Internet

Neste capítulo são apresentados os conceitos e funcionalidades pertinentes à camada internet da arquitetura TCP/IP, tais como sua unidade de informação (datagrama), o funcionamento básico do *Internet Protocol* (IP), definições de endereçamento, roteamento de mensagens e protocolos de mensagens de controle e de erros. Caso o leitor tenha interesse em recordar os conceitos relacionados às arquiteturas de redes de computadores e, em específico, à arquitetura TCP/IP, é sugerida uma leitura prévia no apêndice A. Nos apêndices B e C são encontrados conteúdos complementares à camada Internet, mais precisamente, sobre protocolos de roteamento dinâmico e algoritmos de roteamento, respectivamente.

3.1. Funcionalidades da Camada Internet

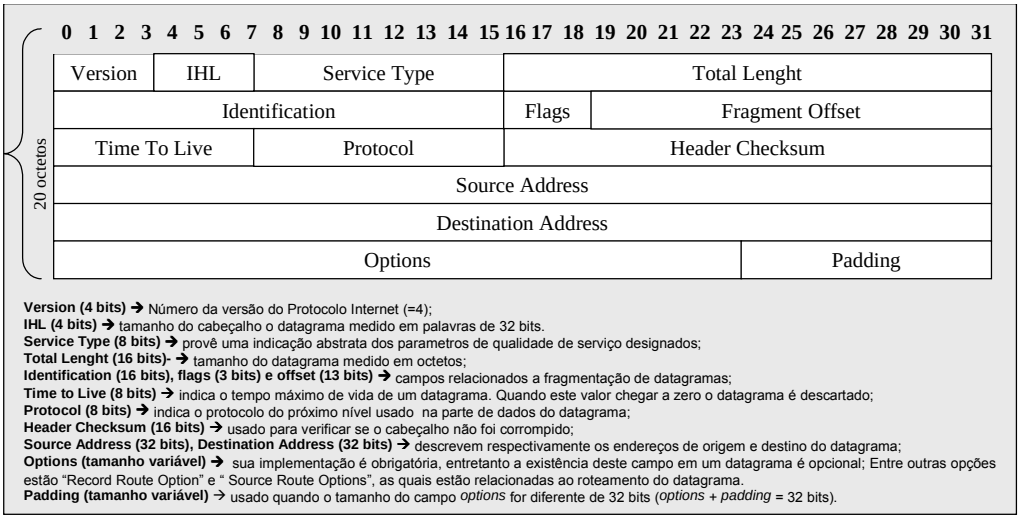
A camada Internet, cujo *"carro-chefe"* é a especificação do *Internet Protocol (IP)*, é responsável por prover a interconexão de redes. As facilidades de se interconectar redes suportadas pelo IP foram e continuam sendo fundamentais para a disseminação da Internet. Os serviços especificados por esta camada não estabelecem conexão nem tão pouco são confiáveis.

Em um **serviço sem conexão** não há um contato prévio entre a origem e o destino. Cada pacote é tratado independentemente dos outros, podendo inclusive serem enviados por caminhos diferentes. Um **serviço de entrega não confiável** significa que pacotes (datagramas) podem ser

perdidos, duplicados, atrasados ou entregues fora de ordem. Portanto, não é função do IP detectar tais condições [Stallings, 98].

O IPv4 [Postel, 81] é a versão do IP mais utilizada atualmente, no entanto, o IPv6 [Deering, 98] é a mais nova versão deste protocolo. O IPv6 foi projetado para acomodar as redes de alta velocidade de hoje e a mistura de *streams* de dados, vídeo, voz, etc., que estão sendo requisitadas cada vez mais. Entretanto, o que provocou o desenvolvimento desta versão foi a questão do esgotamento de endereços disponíveis e o crescimento das tabelas de roteamento. Em ambas as versões, o IP traz uma série de definições, como:

- ✓ **Unidade de Informação:** as especificações do IP definem a estrutura da unidade de informação, o Datagrama (pacote), que irá transitar entre as redes interligadas. Assim, como a maioria das definições de unidade de informação em redes, o Datagrama é composto por um cabeçalho e um corpo da mensagem (a informação em si). O IPv6 traz algumas evoluções do formato do cabeçalho em relação ao IPv4, onde alguns campos do cabeçalho têm sido deixado de lado ou tornados opcionais (figuras 3.1 e 3.2).



Version (4 bits) → Número da versão do Protocolo Internet (=4);
IHL (4 bits) → tamanho do cabeçalho o datagrama medido em palavras de 32 bits.
Service Type (8 bits) → provê uma indicação abstrata dos parametros de qualidade de serviço designados;
Total Lenght (16 bits)- → tamanho do datagrama medido em octetos;
Identification (16 bits), flags (3 bits) e offset (13 bits) → campos relacionados a fragmentação de datagramas;
Time to Live (8 bits) → indica o tempo máximo de vida de um datagrama. Quando este valor chegar a zero o datagrama é descartado;
Protocol (8 bits) → indica o protocolo do próximo nível usado na parte de dados do datagrama;
Header Checksum (16 bits) → usado para verificar se o cabeçalho não foi corrompido;
Source Address (32 bits), Destination Address (32 bits) → descrevem respectivamente os endereços de origem e destino do datagrama;
Options (tamanho variável) → sua implementação é obrigatória, entretanto a existência deste campo em um datagrama é opcional; Entre outras opções estão "Record Route Option" e " Source Route Options", as quais estão relacionadas ao roteamento do datagrama.
Padding (tamanho variável) → usado quando o tamanho do campo *options* for diferente de 32 bits (*options* + *padding* = 32 bits).

Figura 3.1: Cabeçalho do datagrama IPv4 [Comer, 95].

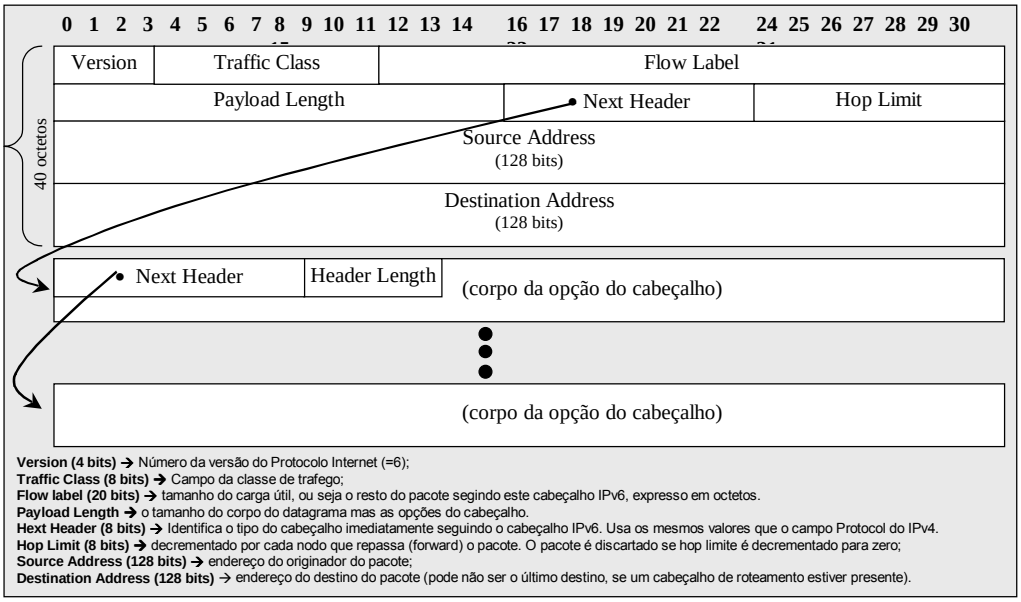


Figura 3.2: Cabeçalho do datagrama IPv6 [Deering, 98b].

- ✓ **Endereçamento:** define o formato e significado de endereços que identificam as interfaces de cada equipamento pertencente a uma rede TCP/IP;
- ✓ **Roteamento de Mensagens:** a principal funcionalidade da camada internet é encaminhar unidades de informação (datagramas) entre redes distintas, desde a origem até o destino. Isto é realizado por um algoritmo (***forward***), que ao receber um datagrama o encaminha para a(s) interface(s) de saída, da forma mais eficaz possível. Estas decisões são tomadas com base em uma **tabela de roteamento**, na qual devem constar as informações pertinentes às melhores rotas para os mais diversos destinos. As tabelas de roteamento são geradas e mantidas pelos **protocolos de roteamento**.
- ✓ **Envio de mensagens de Controle e de Erros:** são mensagens geradas e trocadas entre módulos IP. Esta funcionalidade é implementada pelos protocolos ICMP (*Internet Control Message Protocol*) e IGMP(*Internet Gateway Multicast Protocol*).
- ✓ **Fragmentação e desfragmentação:** para viabilizar a transmissão de datagramas, cujo tamanho for maior do que o MTU[2] (tamanho máximo da unidade de informação) da tecnologia de rede a ser utilizada (ex.: na *Ethernet* o MTU é 1500kb), e
- ✓ **ADD-ONs:** na atualidade, os equipamentos que implementam a camada internet incorporam novas funcionalidades, como regras nos algoritmos de controle de congestionamento e reserva de recursos (ex. RSVP – *ReSerVation Resource Protocol*).

[2] As especificações do IPv6 definem que o MTU da tecnologia utilizada abaixo do IP deve ser de no mínimo 1280 octetos, entretanto é recomendado que se utilize MTU de 1500 octetos para acomodar encapsulamento sem recorrer as funções de fragmentação.

O IP é implementado tanto nos roteadores como nos *hosts* presentes em uma *internetwork*. Este age como um "transmissor" para mover um bloco de dados de um *host*, através de um ou mais roteadores, para outro *host*. Todavia, ao receber um datagrama e analisar o seu endereço de destino (destination address), o *host* e o roteador agem de forma diferente, como por exemplo [Comer, 95]:

- ✓ **Se** endereço-destino do datagrama for diferente (daquele que recebe o datagrama: *host* ou roteador):
  - host descarta o datagrama;
  - roteador executa o algoritmo de roteamento padrão do IP, e então envia para a interface mais indicada;
- ✓ **Se** endereço-destino do datagrama for igual (daquele que recebe o datagrama: *host* ou roteador):
  - host repassa o datagrama para a camada logo acima (transporte);
  - roteador verifica no cabeçalho do datagrama, identificando a qual protocolo da camada internet é destinado o mesmo (ex. ICMP, IGMP, protocolos de roteamento, etc.).

As decisões pertinentes ao encaminhamento dos datagramas são tomadas com auxílio de informações contidas em bases de dados chamadas "tabela de roteamento". Essas tabelas de roteamento são geradas e mantidas por protocolos de roteamento dinamicamente, conforme já foi dito anteriormente. Maiores informações sobre os principais protocolos de roteamento encontram-se no apêndice B.

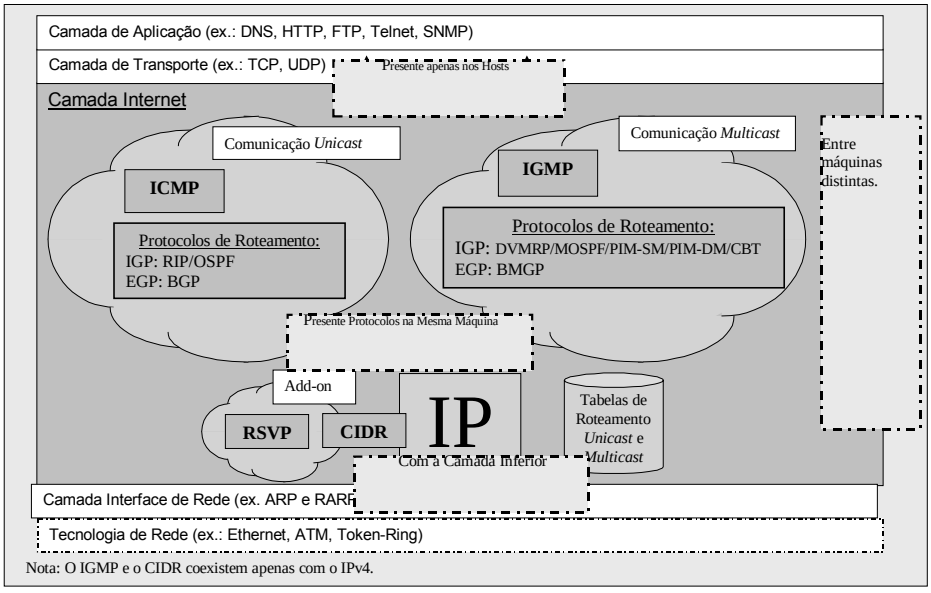


Figura 3.3: Camada Internet da Arquitetura TCP/IP.

A figura 3.3 tem o objetivo de abstrair os principais componentes da camada internet. Pode-se dizer que uma implementação da camada internet tem 3 tipos de interfaces:

- ✓ **Tipo 1 - Interface com a Camada Inferior** (parte inferior da figura 3.3): define a comunicação com a camada Interior (Interface de Rede) da arquitetura TCP/IP.

Estas interfaces definem os enlaces existentes com outros equipamentos (Interface entre Nós da Rede). Um *host* geralmente tem uma única interface deste tipo, a qual o liga a rede local ou diretamente ao roteador. Nos roteadores, existem duas ou mais interfaces. O roteador mantém uma fila de chegada e uma fila de saída de datagramas para cada interface. Do ponto de vista do IP, cada interface é identificada por um endereço IP;

- ✓ **Tipo 2 - Interface com a Camada Superior** (parte superior da fig. 3.3): é o meio pelo qual esta camada recebe e entrega os dados dos usuários (camadas superiores). Teoricamente, este tipo de interface existe apenas nos *hosts*, uma vez que no mundo real, os roteadores requerem algumas aplicações para suportar sessões de *login* (TELNET) remoto e transferência de dados para administração (SNMP) do mesmo [Stallings, 98], através da Camada de Transporte (TCP e/ou UDP);
- ✓ **Tipo 3 - Interface entre Protocolos da Camada Internet:** (centro e lateral direita da figura 3.3): estas interfaces viabiliza a comunicação do protocolo IP com outros protocolos presentes nesta camada (ICMP, protocolos de roteamento e ADD-ONs), tanto no mesmo equipamento, quanto entre equipamentos.

3.2. Endereços Internet

3.2.1. Endereços IPv4

Endereços são fixados em tamanho de quatro octetos (32 bits). Teoricamente o número de endereços disponíveis seria 2^{32}, no entanto, na prática o número de endereços disponíveis é menor, devido à estrutura hierárquica na qual eles estão organizados. A figura 3.4 apresenta a classificação de endereços Internet e a figura 3.5 apresenta a convenção de endereços IPV4.

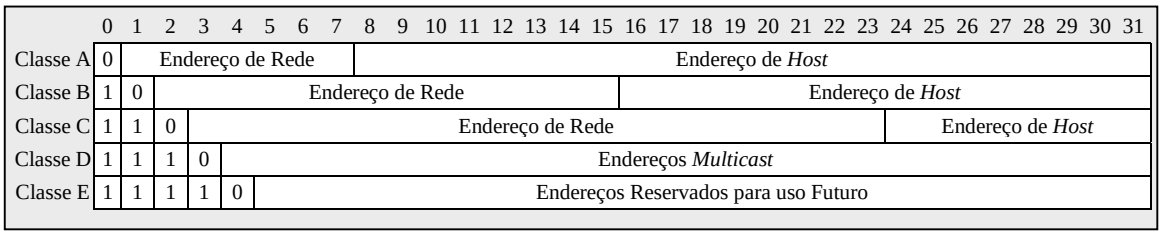


Figura 3.4: As 5 Classes de Endereços Internet [Comer, 95].

Um endereço Internet é divido em duas partes onde, uma é usada para se identificar uma rede (endereço de Rede) e a outra identifica um *host* (endereço de Host) específico dentro da rede. Por convenção, um endereço que tem todos os bits da parte do *host* igual a zero (0) é reservado para se referir à rede. E alguns endereços (fig. 3.5) são definidos como especiais.

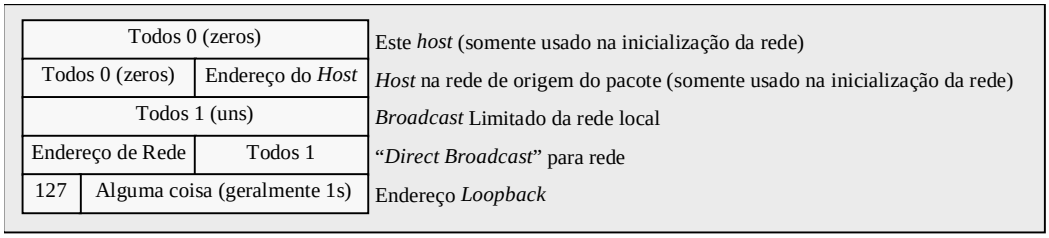


Figura 3.5: Convenção de Endereços IPv4 [Comer, 95].

Existem também alguns endereços *multicast* que têm usuabilidade especial, como mostra a tabela 3.1. Endereços *multicast* não podem ser usados como endereço de *hosts* fonte.

Faixa de Endereços	Usuabilidade (escopo)
224.0.0.1	Todos sistemas
224.0.0.2	Todos roteadores
224.0.0.1 - 224.0.0.255	Usado somente em um segmento local
239.0.0.0 - 239.255.255.255	Escopo de alcance administrativo
239.192.0.0 - 239.195.255.255	Escopo de uma organização local
239.255.0.0 - 239.255.255.255	Escopo local

Tabela 3.1: Atribuições de Endereços *Multicast* [Moy, 98].

3.2.2. Classless Internet Domain Routing – CIDR

O protocolo **IP** tem sido largamente utilizado por mais de uma década, mas apesar de estar funcionando muito bem surgiram dois problemas: a exaustão dos endereços IP e a explosão das tabelas de roteamento. O CIDR é uma solução a curto prazo que consiste, essencialmente, na perda do valor semântico das classes de endereços, passando a informação de roteamento a ser considerada em relação ao par (rede, máscara) ou (rede, n.º de bits significativos) em contraste com a situação original em que a máscara era obtida implicitamente pelas classes de endereçamento [Fuller, 93]. A atribuição de endereços passa a ser feita de uma forma hierárquica, com níveis como provedores de serviços (ex. backbones, redes regionais) e assinantes de serviços (ex. sites, campus, etc.) [Rekhter, 93]. A idéia básica do plano é alocar um ou mais blocos de endereços classe C para cada provedor de serviço da rede. Para cada organização, que se conecte à Internet via provedor, são alocados subconjuntos de endereços deste provedor.

Hipoteticamente, suponha que um provedor **A** tenha 2048 redes classe C que comecem com 192.24.0.0 e terminem com 192.31.255.0. Suponha ainda que um cliente **Z** deseja menos do que 2048 endereços, então serão alocados os endereços 192.24.0.0 a 192.24.7.0 para ele. Esta sub alocação hierárquica de endereços implica que clientes com um subconjunto de endereços de um provedor terão, obrigatoriamente, sua informação transmitida pela infra-estrutura do roteador [Tarouco, 96].

3.2.3. Endereços IPv6

Os endereços IPv6 são constituídos por 128 bits e teoricamente este tamanho pode acomodar 6,65x1023 endereços para cada metro quadrado da terra. Entretanto, assim como o

IPv4, a hierarquia apresentada na estrutura do endereço faz com que a disponibilidade prática de endereços seja menor [Thomas, 96].

Os padrões Internet apresentam um formato para representação dos endereços legíveis aos humanos. O mesmo quebra o endereço em 8 (oito) partes de 16 bits separados por dois pontos(:); estas são expressas em notação hexadecimal (ex. FEDC:BA98:7654:0:FEDC:BA98:7654:3210). Na existência de partes consecutivas com valor zero (FEDC:0:0:0:0:7654:0:3210) pode-se abreviar da seguinte forma: FEDC::7654:0:3210. O IPv6 define 3 tipos de endereços: *Unicast*, *Anycast* e *Multicast*.

O **Endereço *Unicast*** é um endereço que identifica cada interface de um equipamento com outro.

O **Endereço *Anycast*** é um endereço *unicast* que é atribuido para mais de uma interface (tipicamente seguindo para nodos diferentes), o qual é roteado para o destino mais próximo (de acordo com a métrica do protocolo de roteamento) [Deering, 98b]. O endereço *anycast* deve ser explicitado no roteador o qual ele foi atribuído. Este tipo de endereço somente pode ser atribuído para roteadores, e não pode ser utilizado como endereço fonte em um pacote;

O **Endereço *Multicast*** é um identificador de um grupo de nodos. Um nodo pode pertencer a um ou mais grupos *multicast* [Deering, 98c]. A figura 3.6 mostra o formato de um endereço *multicast* . A tabela 3.2 apresenta a descrição do campo Escopo desse endereço.

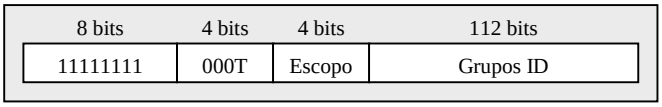


Figura 3.6: Formato do Endereço [Comer, 95].

Na figura 3.6, os 8 primeiros bits (todos setados em um) representam o prefixo do endereço *multicast*. Os 4 bits seguintes são "*Flags*", cujos 3 primeiros bits são 0 e o quarto (T) define se o grupo *multicast* é persistente(0)[3] ou transiente(1)[4]; os próximos 4 bits seguintes definem o escopo do grupo, cujas opções são apresentadas na tabela abaixo [Thomas, 96]:

Valor	Descrição do Escopo
0	Reservado
1	Confinado para um único sistema (segmento-local)
2	Confinado para um único enlace (segmento-local)
5	Confinado para um único site
8	Confinado para um único organização
E	Escopo Global
F	Reservado

Tabela 3.2: Descrição do campo Escopo do endereço *multicast* com seus respectivos valores.

[3] Grupo *Multicast* Persistente é um grupo que sempre existirá.
[4] Grupo *Multicast* Transiente é um grupo temporário, como em uma aplicação de videoconferência esporádica.

O IPv6, assim como o IPv4, associa um endereço a uma conexão de rede específica (interface). Ele também retém (e estende) a hierarquia de endereços IPv4, para isso é atribuído um prefixo para uma rede física. Contudo, para tornar fácil a atribuição e modificação de endereços, o IPv6 permite a atribuição de múltiplos prefixos de tamanho variável (ex.: FF00::/8) a uma certa rede e permite que um computador (ou roteador) tenha múltiplos endereços simultaneamente designados a uma dada interface [Comer, 95]. A tabela 3.3 apresenta uma lista completa dos tipos de endereços que *hosts(H)* e roteadores (R) devem reconhecer.

Tipo de Endereço	Suporte
Um endereço unicast de Link-local para cada interface	H e R
Alguns outros endereços unicast ou anycast* (somente no roteador) atribuidos	H e R
O endereço de *loopback*	H e R
Endereços multicast de "todos os sistamas" (nodo-local e link-local)	H e R
Endereços multicast de "todos os roteadores" (nodo-local e link-local)	R
Endereços multicast do nodo solicitado para todos endereços unicast e anycast	H e R
Endereços multicast para algum outro grupo para o qual o *host*/roteador pertence	H e R
O endereço anycast da SubNet do roteador par cada interface	R

Tabela 3.3: Tipos de endereços a serem suportados por *Hosts* e Roteadores.

A questão da explosão das tabelas de roteamento é tratada com a definição de um número arbitrário de níveis de hierarquias, onde cada uma pode ser representada por um número arbitrário de bits. A figura 3.7 apresenta a hierarquia de endereços Ipv6. Conforme mostra essa figura, os níveis da hierarquia decrescem da esquerda para a direita.

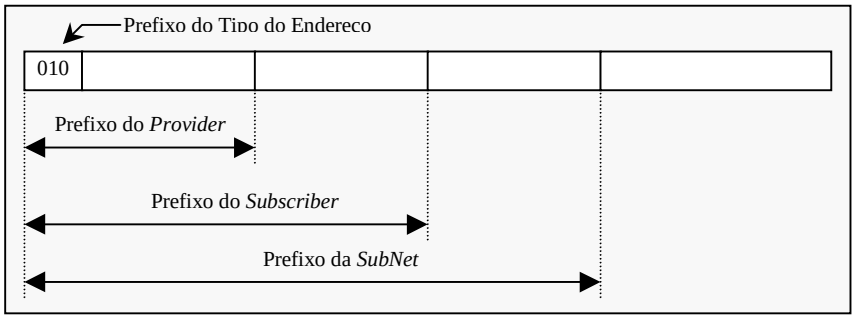


Figura 3.7: Hierarquia de Endereços IPv6 para Provedores de Acesso a Rede [Comer, 95].

O Prefixo do tipo do endereço representa a divisão dos endereços IPv6 em tipos (propostos pelos projetistas do IPv6), os quais são análogos às classes do IPv4. O ***Provider ID*** identifica um provedor de acesso a Rede em específico, cujos clientes são identificados pelo ***Subscriber ID***. Tais clientes identificam suas redes locais através do ***SubNet ID*** e, finalmente, uma interface de um equipamento é identificada pelo ***Node ID***. É recomendado que o ***Node ID*** tenha pelo menos 48 bits para permitir a utilização de endereços definidos pelo IEEE 802 (ex. um endereço *Ethernet*) [Comer, 95].

3.3. Roteamento de Mensagens

3.3.1. *Introdução*

Em uma rede de computadores a escolha do caminho fim-a-fim (rota), isto é, por onde uma mensagem deve transitar é chamada de roteamento [Soares, 95]. Assim, rotear é determinar o melhor caminho (rota) entre uma origem e um ou mais destinos que estejam separados por, no mínimo, um nó intermediário e, enfim, transportar as informações sob este caminho.

Determinar a melhor rota é definir por qual enlace (interface) uma determinada mensagem deve ser enviada, buscando chegar ao seu destino de forma segura e eficiente. Para realizar esta função, o módulo roteador utiliza dois conceitos importantes: o conceito de métrica e o conceito de tabelas de roteamento. Métrica é o padrão de medida usado pelos algoritmos de roteamento para determinar o melhor caminho da origem até o destino. Pode-se utilizar um ou vários parâmetros na composição da métrica. A utilização de vários parâmetros permite uma melhor modelagem da métrica e uma decisão mais eficiente na escolha do melhor caminho. Exemplos de parâmetros: tamanho do caminho (em Km ou nº de saltos), confiabilidade, atraso, largura de banda, carga e custo da comunicação [Schimitt, 96]. As Tabelas de roteamento são estruturas de dados contendo informações dos possíveis caminhos e seus custos, a fim de que se possa decidir qual o melhor. Diversos métodos têm sido utilizados para a manutenção da estrutura de dados [Soares, 95].

Como foi descrito nas sessões anteriores, a função de roteamento pertence à camada de rede (camada Internet na arquitetura TCP/IP), e as implementações desta podem conter um ou vários protocolos de roteamento. A função desses protocolos é construir e manter as tabelas de roteamento completas nos diversos nós de uma rede através da troca de mensagens entre eles.

Uma conjunto de sub-redes é visto como um **sistema autônomo** (figura 3.8) que, geralmente, é gerenciado por uma única entidade, ou pelo menos tem algum grau razoável de técnicas e controle administrativo seguindo as mesmas políticas.

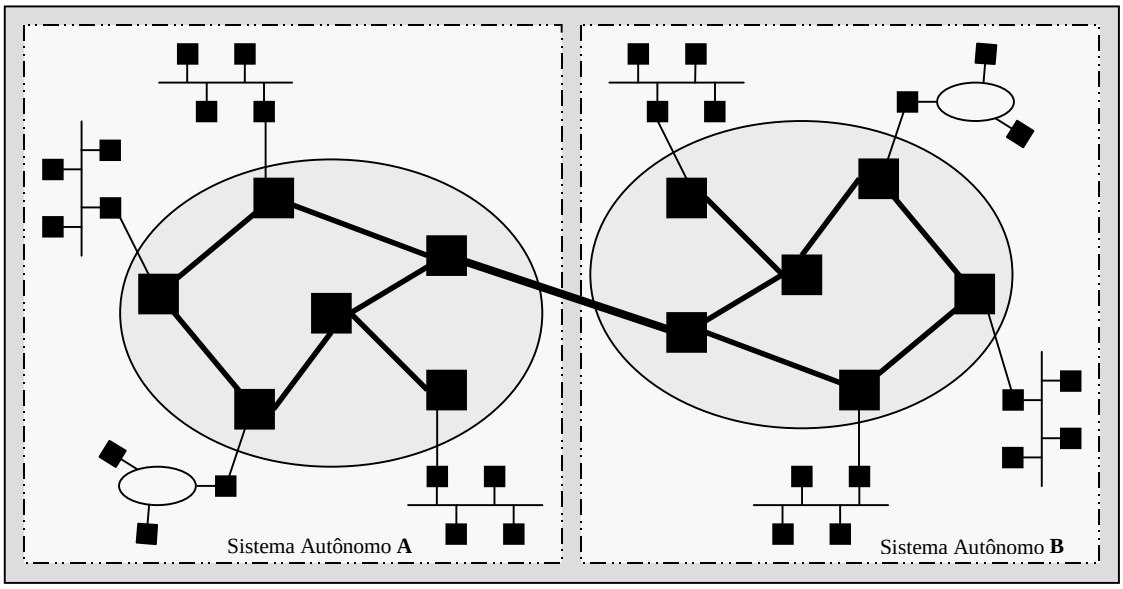


Figura 3.8: Sistemas Autônomos interligados.

Devido à considerável diferença entre rotear informações em um sistema autônomo e entre sistemas autônomos, os protocolos de roteamento, geralmente, são divididos em duas grandes classes:

- ✓ **Interior Gateway Protocols (IGP):** são caracterizados por atuarem nas dependências de um domínio gerencial (sistema autônomo). Os principais objetivos desta classe são descobrir as rotas ótimas e possibilitar a rápida convergência dos nós diante de mudanças (ex. *Routing Internet Protocol* - RIP, *Distance Vector Multicast Routing Protocol* - DVMRP, *Open Shortest Path First* - OSPF, *Multicast Extension OSPF* - MOSPF);
- ✓ **Exterior Gateway Protocols (EGP):** são protocolos que possibilitam a comunicação entre vários sistemas autônomos. Estes sistemas (protocolos) preocupam-se em proporcionar estabilidade e controle administrativo, protegendo um sistema de redes contra erros ou falsificações intencionais por parte de sistemas externos(ex. *Border Gateway Protoco*l - BGP, *Border Multicast Gateway Protocol* - BMGP).

Um protocolo de roteamento pode ser composto por um ou mais algoritmos de roteamento. Os algoritmos determinam as estruturas de dados (Tabelas de Roteamento) e decidem sobre qual enlace um pacote deve ser enviado. Existem vários algoritmos de roteamento, porém todos, com suas características e variantes (ver apêndice B).

Um roteador pode suportar um ou mais protocolos de roteamento. Entretanto a implementação do protocolo IP é obrigatória em todos os roteadores, onde se encontra a funcionalidade ***Forwarding***. Ela corresponde a tarefa de receber um pacote de uma interface e repassá-lo (*forwarding*) para outra(s) interface(s). O *Forwarding* pode ser classificado em *unicast, broadcast* e *multicast*.

3.3.2. Unicast Forwarding

Ao receber um datagrama o protocolo IP executa o algoritmo de roteamento mostrado na figura 3.9, para determinar o que ele deve fazer com o mesmo.

```
Algoritmo ForwardDatagram (Datagrama, TabelaRoteamento)
{
  IPDestino:=ExtrairIP_Destino(Datagrama);
  IPRedeDestino:= ComputarEnderdaRedeIP(IPDestino, PrefixoRedeDestino: enderecoIP);
  Se IPRedeDestino = AlgumaInterfaceLocal()
      EnviaDatagramaNaInterfaceLocal(Datagrama)
 Senão
      Repita
          Se ("E" lógico entre IPDestino e MascaraSubRede) = EntradaNaTabela[i])
             Então
                 EntregaDatagramaInterface(IPProximoSalto)
      Até Que DatagramaEntregue ou FimTabelaRoteamento
      Se FimTabelaRoteamento
          Então
              ICMP_DestinationUnreachable()    /* ErroRoteamento() */
}
```

Figura 3.9: Representação do Algoritmo de Roteamento IP apresentado em [Comer, 95].

3.3.3. Broadcast Forwarding

O *Broadcast Forwarding* consiste em repassar o datagrama recebido para todos os *hosts* presentes na rede. Com o advento IP *multicast*, os endereços *broadcast* e *broadcast direct* têm sido cada vez menos utilizados. Sendo assim, o *broadcast forwarding* não será abordado neste trabalho, entretanto uma descrição sucinta dos tipos de *broadcast* existentes pode ser encontrada em [Moy, 98]. O IPv6 não usa o termo *broadcast* ou *broadcast directed* para se referir à entrega de datagramas para todos computadores em uma rede física ou uma sub-rede lógica. Em vez disto, ele assume que a transmissão *broadcast* é uma forma especial de *multicast* [Comer, 95].

3.3.4. Mulicast Forwarding

Um datagrama *multicast,* identificado pelo endereço destino, é enviado pelo *host* origem, multiplica-se à medida que surgem caminhos diferentes, até chegarem a todos os membros do grupo. Uma rede *multicast* possibilita que uma aplicação envie um único datagrama que será entregue para um ou vários destinos, ou seja, aplicações utilizam comunicações um-para-muitos e muitos-para-muitos.

Serviços *multicast* estão disponíveis na camada de enlace na maioria das tecnologias de LANs, na *ethernet*, onde o endereço MAC (*Meium Access Control*) que tiver o bit menos significativo do 1º byte *setado* em 1 é um endereço *multicast* (ex. 0**1**-00-5e-00-00-01). Os *hosts* ligados a um segmento de rede devem ter seus adaptadores *ethernet* programados para receber pacotes *multicast* [Moy, 97].

O IP *multicast* é uma generalização deste serviço de LAN. Um endereço destino *multicast* refere-se a um grupo *multicast*. Os membros de um grupo *multicast* podem estar espalhados pela

Internet. Um *Host* que deseja enviar um datagrama para um grupo *multicast* não precisa saber quem são nem quantos são os membros deste grupo, e nem mesmo fazer parte do grupo. Os roteadores *multicast* se encarregam de entregar o datagrama para todos os membros do grupo replicando-o quando necessário. A associação a um grupo *multicast* é dinâmica, não há limites de números de membros de um grupo, e um *host* pode fazer parte de mais de um grupo [Moy, 98].

O Tráfego IP *Multicast* para um par particular (Fonte, Grupo Destino) é transmitido da fonte para os receptores via uma *spanning Tree*[5] (árvore de caminhos) que conecta todos os *hosts* no grupo. Diferentes protocolos de roteamento *multicast* usam diferentes técnicas para construir estas *spanning trees multicast*. Uma vez construídas, todo tráfego *multicast* é distribuído sobre elas. Os protocolos de roteamento IP *multicast,* geralmente, podem ser classificados de acordo com o tipo da *spanning tree* [Jonhson, 97]:

- ✓ **Árvore Baseadas na Fonte**: uma árvore é construída para cada combinação de fonte *multicast* (F) e grupo de destino (G). Esta pode ser identificada pelo par (F,G). Após a construção da árvore, existirá exatamente um caminho da fonte para cada destino, e no caso de existirem mais de um caminho de mesmo custo, o algoritmo de construção se encarregará de "podar" caminhos adicionais. A Tabela de roteamento destes algoritmos indexam suas entradas pelo endereços da fonte e do grupo *multicast*. Exemplos são DVMRP, MOSPF e PIM-*Dense Mode*.
- ✓ **Árvore Compartilhada:** é construída uma única árvore que será utilizada por todas as fontes de um grupo existente no domínio de roteamento. Um roteador é eleito como raiz da árvore do grupo (*Rendezvous Point* ou *Core*) e todas as mensagens são enviadas para este roteador, que *forward* para os *host* do grupo. A tabela de roteamento destes algoritmos são indexadas, apenas, pelo endereço do grupo. Exemplos são o PIM-*Sparce Mode* e o CBT.

As tabelas de roteamento geradas pelos protocolos de roteamento *multicast* armazenam a interface de chegada (para controlar a existência de múltiplos caminhos) e de saída para os elementos do grupo em uma entrada.

O processo de *forward datagramas multicast* inicia quando o *host* fonte transmite o datagrama no segmento de rede local como um *multicast* de enlace de dados. O roteador *multicast* local recebe o datagrama, uma vez que ele tem um filtro *multicast* de enlace de dados independente do destino, então executa o seguinte algoritmo mostrado na figura 3.10.

[5] *Spanning T*ree: representação particular de grafos à qual se caracteriza em uma árvore onde existe apenas um caminho entre a raiz (ex. fonte *mulitcast*) e cada uma de suas folhas (ex. receptores *multicast*). Uma *Spanning Tree* garante que a conectividade ente cada par de LAN tenha somente um caminho evitando a existência de *loops*.

```
Algoritmo ForwardDatagramaMulticast (Datagrama)
{
 IPDestino:= ExtrairIP_Destino(Datagrama)
 TabelaR[i]:= Criar_MulticastCacheEntries()*
 Repita
   Se IPDestino = TabelaR.IPMulticast[i]**
     Então {
                Se (IPDestino = TabelaR.InterfaceChegada[i]) e (TabelaR.InterfacesSaída <> Null)
                        Então EntregaDatagrama(Datagrama, TabelaR.InterfacesSaída)
                        Senão DescartaDatagrama(Datagrama)
                                /* há múltiplos caminhos ou não há elementos no grupo*/
            Até que DatagramaEntregue ou Fim TabelaRoteamento
            Se FimTabelaR
                Então ErroRoteamento()
     }
Notas:
     *Entradas na tabela de roteamento também são chamadas de "multicast cache entries", pois
          alguns protocolos de roteamento multicast criam às geram dinamicamente, a
          medida que os datagramas vão chegando;
     ** As tabelas de roteamento armazenam em suas entradas:
          o endereço de chegada (Tabela.InterfaceChegada);
          uma ou mais interfaces de saídas(TabalaR.InterfacesSaida).
```

Figura 3.10: Representação do Algoritmo *Forward Multicast*.

3.4. Envio de mensagens de Controle e de Erros

3.4.1. Internet Control Message Protocol – ICMP

O ICMP, tipicamente, permite que roteadores enviem mensagens de erro ou controle para outros roteadores ou *hosts*. Este protocolo é responsável pela comunicação entre o software IP de uma máquina com o software IP de outra máquina [Comer, 95]. Contudo, um *host* também pode utilizar mensagem ICMP para todas as mensagens de informação e controle. Assim, o uso do ICMP provê um único mecanismo usado para envio de todas as mensagens de informações de controle e erros na camada Internet - IP.

Nome da Mensagem	Usuabilidade
Destination Unreachable error message	Destino não encontrado
Packet Too Big error Message	Tamanho do pacote muito grande
Time Exceeded error message	TTL excedido
Parameter Problem error message	Problemas com parâmetros
Echo Request message	Verifica alcançabilidade de destino
Echo Reply message	Destino confirma que é alcançável
Source Quench	Controle de congestionamento
Address Mask Request	Tratamento das máscara dos endereços
Address Mask Reply	Retorna a mascara da rede
Redirect Message	Usada entre Roteadores e *Hosts*

Tabela 3.4: Principais Mensagens ICMP para IPv4 [Comer, 95].

As mensagens ICMP, ao serem enviadas, são encapsuladas dentro de um datagrama IP. Existem vários tipos de mensagens ICMP, cada uma utilizada em situações pré definidas, conforme mostra a tabela 3.4 [Comer,95].

Embora cada mensagem tenha seu próprio formato, todas elas têm os três primeiros campos do cabeçalho em comum, que são o TYPE (identifica a mensagem), CODE (informações avançadas sobre o tipo da mensagem) e o CHECKSUM (permite verificar se a mensagem não foi corrompida).

É assumido que os roteadores conheçam as rotas ótimas. Os *hosts* iniciam com informações de roteamento mínimas e aprendem novas rotas a partir dos roteadores. As mensagens *Redirect* do ICMP são limitadas às interações entre roteadores e *hosts* diretamente conectados. Os roteadores enviam-nas aos *hosts* quando descobrem que eles não estão utilizando rotas ótimas, para que estes as atualizem.

3.4.2. Internet Group Management Protocol - IGMP

O IGMP [Deering, 98] é o protocolo *multicast* correspondente ao ICMP, o qual implementa a necessidade de comunicação entre *hosts* e roteadores *multicast.* Para que a entrega de datagramas *multicast* tenha sucesso, nenhuma comunicação é necessária entre o *host* de origem do pacote *multicast* e seu roteador do 1º salto. Contudo, um *host* que deseja receber datagramas *multicast* de um determinado grupo *multicast* deve informar seu roteador local de sua associação ao grupo, isto é feito através da mensagem *"IGMP-Host Membership Report"*.

Um *host* envia uma mensagem *"IGMP-host membership report"* em duas situações [Moy, 98]:

- ✓ quando se associa a um grupo *multicast*; e
- ✓ em resposta a uma mensagem *"IGMP-host membership query"*: recebida de um roteador.

Um **Roteador *"Querier"*** é o roteador responsável por enviar, periodicamente, mensagens *"IGMP-host membership query"* ao segmento de rede (rede local) ao qual ele pertence, mantendo contato, dinamicamente, com os membros dos grupos.

Na 1ª versão do IGMP, o roteador *querier* é eleito pelo protocolo de roteamento ativo, entretanto, no IGMPv2 a eleição é feita pelo próprio IGMP. O primeiro roteador que se torna ativo em um segmento de rede assume a função, porém a função de *querier* é repassada para o novo roteador no sistema, caso o mesmo tenha um endereço IP menor (o roteador de menor endereço IP é eleito *querier*).

Mensagens *Host membership query* são enviadas para todo o sistema através do endereço 224.0.0.1, e uma resposta (*host membership report*) dos *host* é enviada para cada grupo (no roteador mais próximo) o qual ele pertence. Caso não ocorrer nenhuma resposta correspondente a um determinado grupo, é assumido que não existe membros deste grupo no segmento de rede. Desta forma, os roteadores podem cessar o envio de datagramas do grupo para este segmento.

Para minimizar o nº de *IGMP-host membership report* na rede, um *host*, ao receber uma mensagem *IGMP-host membership query*, aguarda um período de tempo aleatório e caso chegue uma *IGMP-host membership report* (de outro *host*) correspondente ao mesmo grupo, ele pode cancelar o envio de sua resposta.

É importante ressaltar que os roteadores *multicast* não estão interessados em quantos elementos de um grupo constam em um segmento de rede. Eles objetivam saber se existe ou não elementos do grupo neste segmento, pois todos elementos de um segmento de rede compartilham o mesmo meio de comunicação, onde as informações são enviadas por *broadcast* ou *multicast* pela camada de enlace. Redes que não são *broadcast* devem simular as características *multicast* na camada de enlace de dados para rodar o IGMP (ex. LAN *Emulation*).

Na 2ª versão, o IGMPv2 insere um novo mecanismo para diminuir o tempo entre o instante em que o último elemento de um grupo o deixou e o instante em que os roteadores *multicast* deste grupo descobrem o acontecido (latência de saída). Para tal, O IGMPv2 criou um novo tipo de pacote - *"IGMP-Leave Group"*, o qual é enviado pelo *host,* que deseja deixar um grupo, para todos roteadores do segmento de rede (endereço 224.0.0.2). Ao receber o *IGMP-Leave Group*, o *Querier* do segmento de rede *multicast* envia uma mensagem *IGMP-membership query* especifica para o grupo em questão (com TTL = 1), se ninguém responde então sabe-se que este grupo não tem mais elementos neste segmento.

A 3ª versão do IGMP (em desenvolvimento) permite que membros de um grupo filtrem suas fontes, utilizando um novo "Relatório de Associados". Quando junta-se a um grupo, o *host* pode requisitar o recebimento de datagramas *multicast* apenas de fontes específicas (*IGMP-source-specific joins*) ou o não recebimento de datagramas *multicast* de determinadas fontes (*IGMP-source specifics leaves*).

As principais mensagens definidas pelo IGMP são [Moy, 98]: *Host Membership Query, Host Membership Report, IGMPv2 Membership Report, IGMPv2 Leave Group Message, IGMPv3 Membership Report, Multicast Traceroute Query / Request, Multicast Traceroute Response, DVMRP Packets*

3.4.3. Internet Control Message Protocol para IPv6 – ICMPv6

O ICMPv6 é usado por *hosts* e roteadores para relatar erros encontrados no processamento de datagramas e para executar outras funções da camada internet, como descoberta de vizinhos, auto-configuração de endereços (sendo usado pelo DHCP – *Dynamic Host Configuration Protocol*), gerenciar grupos *multicast* e diagnósticos da rede (ICMPv6 "*ping*"). O ICMPv6 é uma parte integral do IPv6 e deve ser completamente implementado em cada equipamento IPv6 [Conta, 98]. Uma mensagem ICMPv6 é enviada pelo uso de uma extensão do cabeçalho do datagrama IP (código 58 do campo *Next Header*).

O ICMPv6 incorpora as funcionalidades que o IPv4 atribui ao IGMP, portanto o IPv6 não se utiliza do IGMP para manipular mensagens *multicast*. Afora isto, o ICMPv6 é muito semelhante ao ICMP, inclusive quanto ao formato da mensagem, no que condiz aos 3 primeiros campos do cabeçalho (*Type*, *Code* e *Checksum*). A seguir, na tabela 3.5, são apresentadas algumas mensagens ICMPv6 e possíveis utilizações das mesmas:

Nome da Mensagem	Usuabilidade
Destination Unreachable error message	Destino não encontrado
Packet Too Big error Message	Tamanho do pacote muito grande
Time Exceeded error message	TTL excedido
Parameter Problem error message	Problemas com parâmetros
Echo Request message	Verifica alcançabilidade de destino
Echo Reply message	Destino confirma que é alcançável
Group Membership Query	Utilizada para comunicação *Multicast*
Group Membership Report	Utilizada para comunicação *Multicast*
Group Membership Termination	Utilizada para comunicação *Multicast*
Router Solicitation	Pertinente ao roteamento
Router Advertisement	Pertinente ao roteamento
Neighbor Solicitation	Descoberta de vizinho
Neighbor Advertisement	Descoberta de vizinhos
Redirect Message	Usada entre Roteadores e *Hosts*

Tabela 3.5: Principais Mensagens ICMPv6.

3.5. Estratégias para Migração do IPv4 para IPv6

A Internet hoje continua utilizando o IPv4, mas já estão sendo instaladas sub-redes com base no IPv6, principalmente no mundo acadêmico. Como é impossível a realização de uma transição instantânea do IPv4 para o IPv6, estão sendo estudadas e testadas estratégias de transição entre estes protocolos. A grande preocupação é manter a compatibilidade entre ambos, tanto nos roteadores quanto nos *hosts*.

Uma gama de ferramentas (mecanismos) estão sendo estudadas e implementadas e novas certamente irão surgir (o Anexo D mostra uma breve discussão dos *drafts* propostos na IETF). Alguns exemplos são definidos em [Gilligan, 00]:

- ✓ **Camada IP Dupla (ou Pilha Dupla):** consiste em uma técnica para prover suporte completo para ambos os protocolos internet (IPv4 e IPv6) nos *hosts* e nos roteadores;
- ✓ **Tunneling Configurado do IPv6 para IPv4**: provê túneis ponto-a-ponto pelo encapsulamento de pacotes IPv6 dentro de cabeçalhos IPv4 para transmiti-los sobre uma infra-estrutura IPv4;
- ✓ **Endereços IPv6 compatíveis com IPv4**: um formato de endereço IPv6 que embutem endereços IPv4, e
- ✓ **Tunelamento Automático de IPv6 sobre IPv4**: Um mecanismo usando endereços IPv4 compatíveis para encapsular datagramas IPv6 automaticamente sobre redes IPv4.

A decisão de quais ferramentas serão utilizadas para transição é tomada de acordo com as necessidades específicas de implementações e sites particulares [Gilligan, 00].

Capítulo 4

4. Especificação de Componentes de Redes TCP/IP

Este capítulo apresenta uma especificação de componentes voltada para a modelagem de redes TCP/IP. Os componentes se apresentam genéricos de forma a representar as funcionalidades mínimas dos elementos básicos de uma rede TCP/IP, os quais são: fontes de tráfego (Aplicação gerando tráfego), *hosts*, enlaces, roteadores e sorvedouros (Aplicação recebendo tráfego). Essa especificação foi feita utilizando a Linguagem de Modelagem Unificada (UML) enfocando principalmente a fase planejar e elaborar, a fase de análise, e a fase de projeto de alto nível (projeto arquitetural e projeto detalhado dos componentes).

A seção 4.1 apresenta considerações sobre a arte de documentar. A seção 4.2 descreve o domínio do problema apresentando os requisitos levantados e os diagramas de *use-cases* que descrevem as interações entre usuários e o sistema. A seção 4.3 descreve a fase de análise apresentando diagramas de conceitos e diagramas de seqüência relevantes para o estudo apresentado. A seção 4.4 apresenta um modelo de alto nível para a arquitetura do sistema, identificação de pacotes, e descrição de componentes de classe em baixo nível. Finalmente, a seção 4.5 apresenta um glossário dos termos relevantes.

4.1. Introdução

4.1.1. A Arte de Documentar

O propósito de usar uma linguagem ilustrada abstrata para as fases de análise e projeto, em um processo de desenvolvimento de software, é que se pode declarar o que é mais importante de forma clara. Cada assunto pode ser dividido no nível de detalhamento que se deseja, sem obscurecer o documento com muitos detalhes. A arte de uma boa documentação é a arte de omitir informações irrelevantes, permitindo que o documento fique bem estruturado, conciso e legível. Um bom documento pode intercalar notações informais, semi formais e formais, sob forma de narrativas, figuras, diagramas, tabelas, etc.. Também é interessante que se elabore um glossário dos termos mais representativos. Quanto à estrutura do documento, é interessante que se ofereçam visões internas e externas separando diferentes aspectos de uma área de interesse [D'Souza, 98].

Em um projeto de desenvolvimento de software geralmente se elaboram 3 documentos, os quais representam diferentes níveis de abstração[D'Souza, 98]:

- ✓ Modelo do Negócio: o "Lado de Fora". Uma implementação descrevendo a figura dos usuários no mundo onde ele trabalha: quais conceitos, como eles são relatados, e como eles interagem. O negócio aqui significa uma parte do mundo no qual se está interessado, incluindo sistema de software;
- ✓ Especificação de Componentes: o "limite". A Especificação do comportamento e a estrutura de um componente, sem abordar o projeto interno. O documento pode mostrar colaborações entre o componente e os objetos em torno dele. Isto freqüentemente é estruturado em diferentes áreas de assunto, as quais enfocam diferentes aspectos do sistema;
- ✓ Projeto do Componente: o "Interior". Implementação mostrando como os componentes trabalham, detalhando como responsabilidades são distribuídas no interior dos componentes

Estes níveis são encontrados (com vários nomes) na maioria dos livros sobre orientação a objetos. Na prática, pode-se utilizar somente algum destes níveis ou introduzir outros níveis entre os mesmos [D'Souza, 98]. Este trabalho empenha-se na realização da documentação dos dois primeiros níveis de abstração citados acima, porém sem se deter a essas terminologia.

No contexto deste trabalho, "especificação" entende-se como sendo a documentação gerada como um todo, a partir da descrição dos componentes propostos. Essa documentação é dotada de figuras, tabelas e artefatos UML, devidamente amparados por narrativas explicativas. Para cada artefato UML utilizado é feita uma breve conceituação e justificativa da utilização do mesmo. Este documento é apresentado em quatro partes: Domínio do Problema, Fase de Análise e Fase de Projeto (Arquitetural) e Glossário de Conceitos Relevantes.

4.2. Domínio do Problema

O objetivo deste trabalho é apresentar uma especificação de componentes voltados para a modelagem de redes TCP/IP. Os componentes propostos são genéricos de forma a representar as funcionalidades mínimas dos elementos básicos de uma rede TCP/IP, os quais são fontes de tráfego, *hosts*, enlaces, roteadores e sorvedouros. Essa especificação de componente poderá ser reutilizada em projetos de ambientes de simulação orientados a componentes voltados para a modelagem e avaliação de desempenho de redes de computadores, como também, poderá ser reutilizada em ambientes de simulação de alto nível, no qual o Arena é um exemplo, permitindo que novos componentes sejam a eles integrados.

O conhecimento da camada internet, no contexto deste capítulo, é imprescindível. Descrições sobre as funcionalidades e características básicas pertinentes à camada internet foram apresentadas no capítulo 3.

Os componentes a serem especificados, representam os principais elementos de uma rede TCP/IP de longa distância. Uma ou mais aplicações de usuários podem se conectar através de *hosts*[6] a uma rede pública (*backbone*), que é formada por um conjunto de roteadores interconectados, como apresentado na figura 4.1.

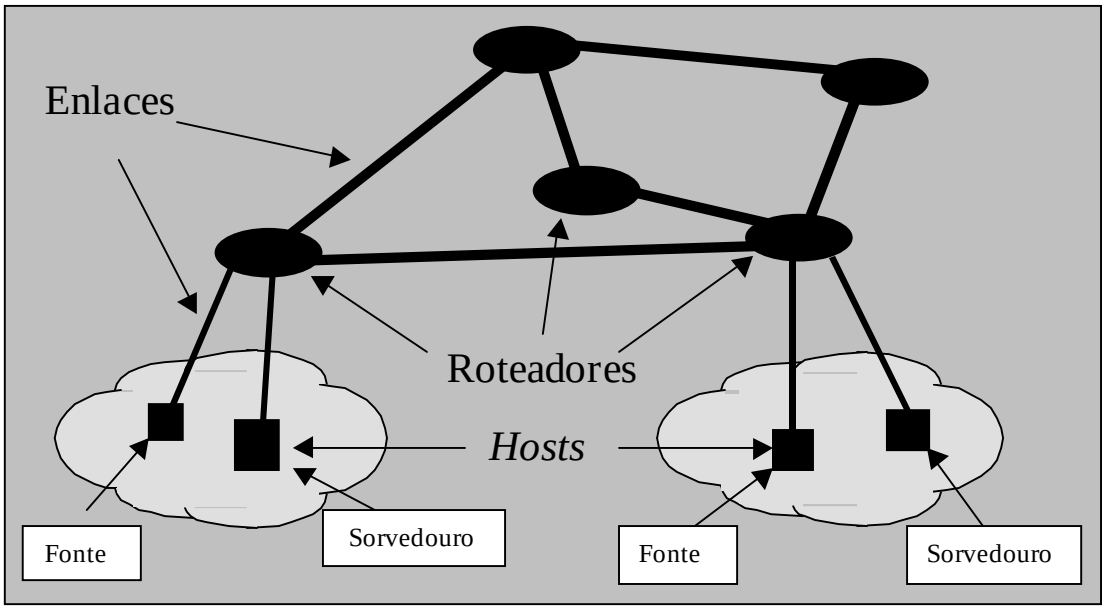


Figura 4.1: Exemplo de cenário de uma rede TCP/IP.

As aplicações de uma rede TCP/IP geram tráfego de informações, as quais devem ser representadas por componentes "fontes de tráfegos". Estas informações serão encaminhadas para os *hosts* de origem, onde serão montadas as unidades de informações, denominados datagramas (pacotes). Devido à existência de aplicações com características de tráfego diversas (texto, voz, vídeo, etc.), uma fonte de tráfego deve representar um único tipo de tráfego.

As informações geradas nas fontes de tráfego, trafegam pela rede através dos roteadores, até o destino. Um *host* destino, ao receber datagramas, simplesmente os repassa a um elemento denominado sorvedouro, o qual registra algumas medidas de desempenho e descarta o datagrama. Esse elemento representa o recebimento do datagrama pela camada superior à camada IP.

Durante esse percurso, devem ser coletadas diversas medidas de desempenho nos componentes da rede. Essas medidas correspondem aos atrasos sofridos pelos datagramas ao longo desse percurso (em cada enlace, em cada roteador); ao número de datagramas recebidos e descartados em cada roteador; ao comprimento de filas dos roteadores, e ao número de datagramas enviados/recebidos pelos *hosts*.

4.2.1. Usuário Alvo

Os documentos resultantes deste trabalho servirão de auxílio para usuários de dois perfis distintos: desenvolvedor de software de simulação e o analista de modelagem de redes de computadores. O primeiro poderá utilizar-se deste documento para implementar os componentes apresentados, podendo estender as funcionalidades de ferramentas de simulação existentes ou facilitando a construção de novas ferramentas para a modelagem e a avaliação de modelos de redes de computadores. Já o analista de modelagem poderá utilizar-se das especificações deste documento para desenvolver simuladores de redes de computadores voltados as suas necessidades específicas, podendo utilizar ou não uma ferramenta de simulação como suporte.

4.2.2. Metas para Solução do Problema

A meta básica da especificação apresentada neste capítulo é a sua reutilização em processos de desenvolvimento de software para simulação de modelos de computadores baseadas na tecnologia TCP/IP. Para alcançar essa meta, torna-se necessários definir metas mais específicas, as quais são:

- ✓ Definir um nível de abstração do funcionamento da redes baseadas na tecnologia Internet;
- ✓ Identificar os elementos básicos existentes neste nível de abstração;
- ✓ Identificar relacionamentos existentes entre esses elementos;
- ✓ Especificar componentes que representam tais elementos.

4.2.3. Levantamento de Requisitos

Os requisitos são uma descrição das necessidades ou desejos para um produto. O principal objetivo desta fase é identificar e documentar o que realmente é necessário. O

[6] Este trabalho considera apenas enlaces ponto-a-ponto, não considerando então, topologias física multiponto, característica de redes locais (ex. *Ethernet*).

documento resultante será um meio de comunicação entre o usuário e o analista de sistemas (desenvolvedor) [Larman, 98]. Os requisitos do sistema podem ser classificados como Funcionais (o que o sistema deve fazer) e Não-Funcionais (atributos do sistema). Os requisitos funcionais se dividem em categorias [Larman, 98], conforme apresenta a tabela 4.1.

Categoria de Funcionalidade	Significado
Evidente	Usuário está ciente de que a função está sendo feita
Escondida	Embora a função seja feita, ela é invisível ao usuário. Tal funcionamento freqüentemente é esquecido ao levantar requisitos
Opcional	Funcionalidade opcional; sua adição não afeta outras funções ou custo significativamente.

Tabela 4.1: Categorias de funcionalidades dos requisitos funcionais[Larman, 98].

Os requisitos não funcionais, também chamados de atributos do sistema, especificam restrições impostas à solução, como aquelas relacionadas com a adoção de padrões e integração com outros sistemas: flexibilidade, facilidade de uso, portabilidade, etc [Freire, 00].

A seguir, na tabela 4.2, são apresentados os requisitos Funcionais (F) em suas respectivas categorias (E=Evidente, H=Escondido, O=Opcional) e Não-Funcionais (NF) pertinentes à abstração de componentes de uma rede TCP/IP.

Ref.	Funcionalidade	Cat.
F1	Os componentes devem representar os elementos básicos de uma *internetwork* (roteador, enlace, *host*, fonte de tráfego, sorvedouro).	E
F2	Os componentes devem fornecer as funcionalidades mínimas dos elementos considerados em F1.	E
F3	Um roteador deve conter 2 ou mais interfaces.	E
F4	As funcionalidades mínimas requeridas ao roteador são: (a) receber datagramas (dados e informações de controle); (b) executar o algoritmo *forward* (*unicast* e *multicast*), o qual consulta a tabela de roteamento; (c) repassar os datagramas pelas interfaces de saída correspondentes; (d) atualizar tabela de roteamento.	E
F5	Prover troca de mensagens de controle entre os componentes (*host*→Roteador→Roteador→*host*). Esta funcionalidade é implementada pelos protocolos ICMP e IGMP.	O
F6	O roteador deve prover comunicação *unicast* ou *multicast*.	E
F7	A topologia inicial deve ser configurada pelo usuário;	E
F8	Todos os componentes são configuráveis pelo usuário. A configuração do roteador consiste na escolha de protocolos (RIP, OSPF, MOSPF, etc.), tempo de processamento de um datagrama, definição das interfaces, etc. A configuração dos enlaces consiste na definição de sua capacidade.	E
F9	Ao receber dados da fonte de tráfego, o *host* deve montar os datagramas, e os encaminhar ao roteador. Ao receber dados da rede (enlaces), o *host* simplesmente deve os repassar ao sorvedouro.	E
F10	Cada *host* deve conter duas interfaces: uma para conectá-lo ao enlace e a outra para conectá-lo a uma suposta aplicação, a qual é representada por fontes ou sorvedouros, respectivamente se este *host* é origem ou destino de uma comunicação.	E
F11	Uma interface deve permitir a entrada e saída de dados em um componente. Cada interface (nos *hosts* e roteadores) deve conter uma fila de entrada e uma fila de saída de dados.	E
F12	O componente “fonte de tráfego” deve gerar tráfego conforme uma aplicação.	E
F13	Roteadores devem suportar a estratégia de transição do IPv4 para IPv6 “Camada Dupla” e ser extensível para implementação de outras estratégias.	O

Ref.	Funcionalidade	Cat.
F14	Os componentes devem permitir à coleta de medidas de desempenho mais relevantes, entre elas: atraso médio de pacotes, quantidade de pacotes descartados, utilização dos enlaces, tamanho médio (e máximo) de filas. As medidas de desempenho são coletadas no decorrer da simulação	H
F15	Os roteadores e enlaces devem suportar mecanismos que os ativam/desativam em tempo de execução (configurados pelo usuário). Esta funcionalidade servirá para permitir alterações na topologia da rede.	E
F16	O componente "sorvedouro" deve receber os datagramas dos *hosts,* coletar algumas medidas de desempenho e descarta-los.	E
NF1	Flexibilidade: possibilidade de agregação de novas funcionalidades aos componentes existentes.	-
NF2	Facilidade de Uso: refere-se a facilidade de configuração dos componentes da rede por parte do usuário.	-
NF3	Documentação: os documentos devem ser gerados seguindo uma linguagem padrão de modelagem (UML), para permitir sua utilização nas 3 primeiras fases da metodologia para estudos de simulação, descrita no capitulo 2.	-
NF4	Compatibilidade: as especificações dos componentes, uma vez que seguem os princípios da análise e projeto de sistemas orientados a objetos, serão melhores aproveitadas na utilização de linguagens de simulação e de propósito geral e em ambientes de simulação de alto nível que incorporam os princípios da orientação a objetos. Entretanto nada impede que seja utilizada em projetos que utilizam linguagens não orientadas a objeto.	-

Tabela 4.2: Descrição dos Requisitos Funcionais e Não-Funcionais.

4.2.4. Diagrama Use-Case

Um diagrama *use-case* descreve graficamente como ocorre uma interação típica entre um usuário e um sistema. Cada um dos *use-cases* descritos abrange uma função a ser disponível ao usuário e define um objetivo de projeto [Larman, 98]. Desta forma, este diagrama é bastante importante durante a fase de especificação de requisitos, onde a funcionalidade que o sistema deverá apresentar deve ser descrita. Um diagrama *use-case* é composto por Atores (entidade externa ao sistema) e *use-cases* (acontecimentos que ocorrem no sistema).

Os componentes devem estar inseridos em um ambiente de simulação. O analista de modelagem, por intermédio de uma interface, proporcionada pelo ambiente de simulação, terá condições de configurar os componentes de acordo com a finalidade de sua avaliação. O diagrama de *use-case*, apresentado na figura 4.2, estabelece as interações dos componentes com o mundo externo.

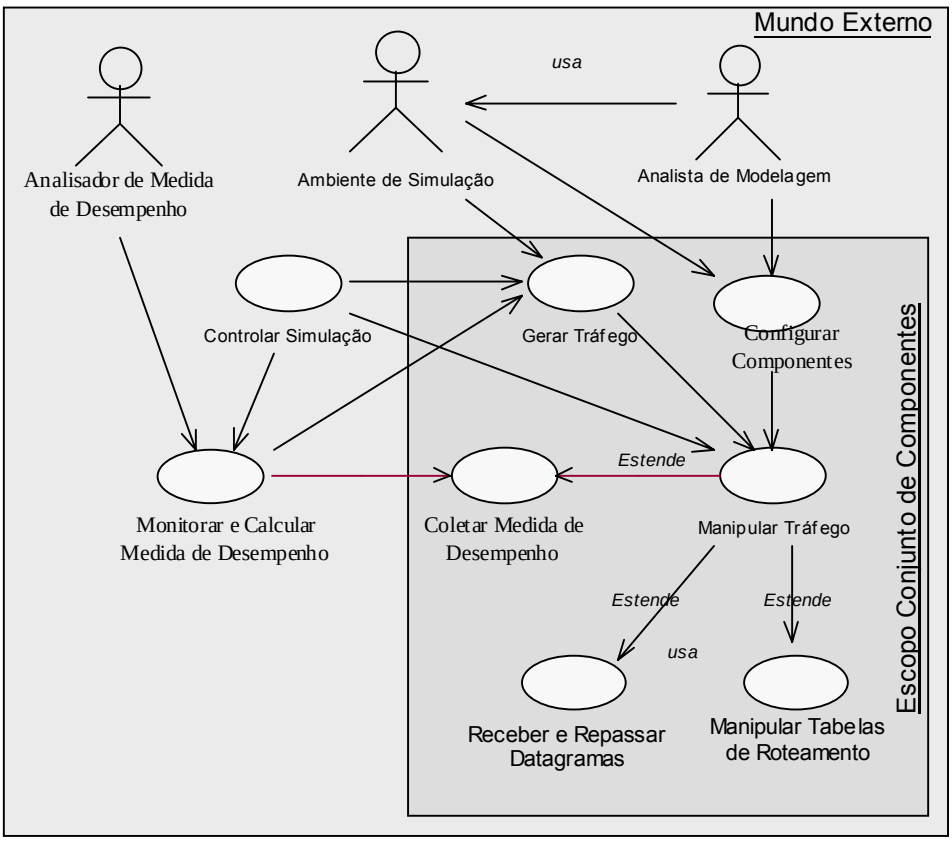


Figura 4.2: Diagrama *use-case* a Camada IP.

Segue uma descrição dos *use-cases,* observando que os *use-cases Controlar Simulação"* e *"Monitorar e Calcular Medidas de Desempenho"* se situam no que se denominou "mundo externo", não pertencendo, no entanto, ao escopo dessa Dissertação.

Use-case: Controlar a Simulação;

Atores: Ambiente de Simulação;

Descrição: O Ambiente de Simulação controla a simulação como um todo, desde proporcionar a interface (gráfica, de preferência) ao Analista de Modelagem para que este gere e configure seus modelos, até controlar o tempo de simulação e o escalonamento de eventos.

Use-case: Monitorar e Calcular Medidas de Desempenho;

Atores: Analisador de Medidas de Desempenho;

Descrição: O Analisador de Medidas de Desempenho coleta e calcula as informações referentes às medidas de desempenho relevantes, durante todo o decorrer da simulação.

Use-case: Configurar Componentes;

Atores: Analista de Modelagem, Ambiente de Simulação;

Descrição: O Analista de Modelagem, através de uma interface proporcionada pelo Ambiente de Simulação, configura cada componente do modelo (sendo trabalhado), como interfaces e protocolos que farão parte do mesmo.

Referências Cruzadas: F1, F3, F7, F8 e F15;

Use-case: Gerar Tráfego;

Atores: Ambiente de Simulação;

Descrição: O Ambiente de Simulação dispara uma fonte de tráfego, cujas informações (*streams*) são entregues ao *host,* por intermédio de sua interface, diretamente conectado, o qual encapsula as informações em datagramas. Medidas de desempenho devem ser coletadas durante a geração de tráfego.

Referências Cruzadas: F1, F9, F10, F11, F12 e F14.

Use-case: Manipular Tráfego;

Atores: não há interação direta com atores;

Descrição: Esta atividade corresponde a todo processamento de roteamento e transmissão de datagramas entre os componentes de uma rede TCP/IP. Este *use-case* é estendido nos *use-cases* explicados a seguir.

Referências Cruzadas: F1, F1, F3, F4, F5, F6, F11, F2, F14 e F16;

Use-case: Receber e Repassar Datagramas

Atores: não há interação direta com atores;

Descrição: Ao iniciar a execução da simulação, a atividade “Receber e Repassar Datagramas” (protocolo IP) é acionada em todos os *hosts* e roteadores que compõem o modelo. Esses componentes podem usar os protocolos ICMP/IGMP para prover a comunicação entre eles. Quando um *host* ou roteador recebe um datagrama de dados, é consultada uma tabela de roteamento (criada pelo protocolo de roteamento dinâmico) para informar sobre o seu repasse (*forward*) ao próximo elemento da rede (*host* ou roteador), seja *unicast* ou *multicast*. O IP mantém-se ativo até que a simulação seja finalizada;

Referências Cruzadas: F1, F3, F4, F5, F6, F9, F11, F12, F5 e F16.

Use-case: Criar e Manter Tabelas de Roteamento;

Atores: não há interação direta com atores;

Descrição: Assim que a simulação é iniciada, o protocolo de roteamento dinâmico (*unicast* e *multicast*) inicia a tarefa de criar as tabelas de roteamento do componente Roteador. No decorrer da simulação os protocolos de roteamento mantém-se ativos no intuito de atualizar as tabelas de roteamento. Em uma determinada simulação, é necessário que um único protocolo de roteamento IGP (*Interior Gateway Protocol*) *unicast*

esteja processando em cada roteador e, opcionalmente, um IGP *multicast*, de acordo com o interesse do Analista de Modelagem;

Referências Cruzadas: F1, F2, F3, F4, F5, F6, F7, F8, F11, F14 e F15;

Use-case: Coletar Medidas de Desempenho;

Atores: não há interação direta com atores;

Descrição: Cada componente coleta medidas de desempenho referentes as suas particularidades, durante todo o decorrer da simulação

Referências Cruzadas: F1, F8, F11, F12, F14 F15 e F16;

Sugere-se que os principais *use-cases* devem ser expandidos para melhor entendimento do processo [Sauvé, 00]. O *use-case* "Manipular Tráfego" foi estendido em mais três *use-cases*: "Receber e Repassar Datagramas", "Criar e Manter Tabelas de Roteamento" e "Coletar Medidas de Desempenho". Devido a esse conjunto de *use-cases* não interagirem diretamente com os atores do sistema, e provocarem uma imensa interação entre os elementos que farão parte dos componentes, a própria descrição da camada internet (capítulo 3) será utilizada com maior detalhamento dos mesmos.

4.3. Fase de Análise

Esta fase de desenvolvimento, enfatiza uma compreensão dos requisitos, dos conceitos e das operações relacionados com um sistema. Investigação e análise são freqüentemente caracterizadas por focalizarem questões do tipo ***qual*** – quais são os processos, os conceitos, os eventos e as operações [Larman, 99]. O principal documento gerado pela análise é um modelo conceitual, o qual facilita a compreensão do domínio do problema e trata, em alto nível, de como uma possível solução pode ser montada para atender aos requisitos levantados anteriormente [Sauvé, 00].

Uma técnica adotada para se gerar o modelo conceitual, é isolar os substantivos das descrições textuais dos *use-cases*, e então utilizar algumas regras encontradas em [Rumbaugh, 94] e [Sauvé, 00] para selecionar os conceitos relevantes. O próximo passo é adicionar associações entre os conceitos. Finalmente, adicionam-se atributos aos conceitos.

Na tabela 4.3 são apresentados os substantivos candidatos a conceitos:

Funcionalidade da Camada IP				
Camada Internet	Origem	Endereço	Roteadores	Alg. controle de congestionamento
IP	Destino	Interfaces	Internetwork	Interface com as camadas inferiores
IPv4	Pacotes (datagramas)	Interfaces de saída	ICMP	Interface com as camadas superiores
IPv6	Protocolo	Mensagens de Erro	IGMP	Mensagens de Controle
Host	Versão	Interconexão de redes	Fragmentação	Serviço de entrega não confiável
Redes	Especificações	Rede TCP/IP	Desfragmentação	Camada de Transporte
Internet	Unidade de Informação	Tabela de Roteamento	MTU	Cabeçalho do Datagrama
RSVP	Campos do Cabeçalho	Protocolo de Roteamento	Endereço destino	
ADD-On	Endereçamento	Serviço sem Conexão		
Endereços Internet				

Endereço IP	Endereço Multicast	CIDR	Endereços IPv6	Endereço Anycast
Endereço de Rede	Endereços Especiais	Classes de Endereços	Endereço Unicast	Prefixo do endereço Multicast
Endereço de Host	Endereço Loopback	Blocos de Endereços	Endereço Multicast	Grupo Multicast
Roteamento de Mensagens				
Roteamento	Entidade	Serviço multicast	CBT	Membro do Grupo Multicast
Tabelas de Roteamento	Protocolo de Roteamento	Grupo multicast	DVMRP	Árvore Baseada na Fonte
Métrica	Algoritmo de Roteamento	IP Multicast	PIM-DM	Tabela de Roteamento Multicast
Origem	Enlace	Fonte Multicast	PIM-SM	Endereço Fonte do Grupo Multcast
Destino	Forward	Tráfego IP Multicat	MOSPF	Rendezvous Pont (core, raiz da árvore)
Sistema Autônomo	Forward Multicast	Spanning Tree		
Internet Control Message Protocol – ICMP		Estratégias de transição do IPv4 par aIPv6		
Comunicação Multicast		Tunelamento	Tunelamento Automático	Mensagem IGMP
ICMPv6		Camada IP Dupla	Endereços	Mensagem ICMP
Mensagem de Erro		ICMP	Comatíveis	Mensagem de Controle

Tabela 4.3: Substantivos candidatos a conceitos.

4.3.1. Diagrama de Conceitos

Conforme pode-se observar nos dados mostrados na seção anterior, levantou-se um grande número de candidatos a conceitos, no contexto deste trabalho, ressaltando a abrangência do domínio do problema. Um estudo inicial, levou a uma proposta de um diagrama abrangente, englobando muitos dos candidatos a conceitos apresentados. Esse diagrama, denominado "Diagrama de Conceitos Abrangente", mostrado na figura 4.3, ressalta, naturalmente, a grande quantidade de interações entre esses conceitos e já evidencia a complexidade de modelagem do sistema em questão.

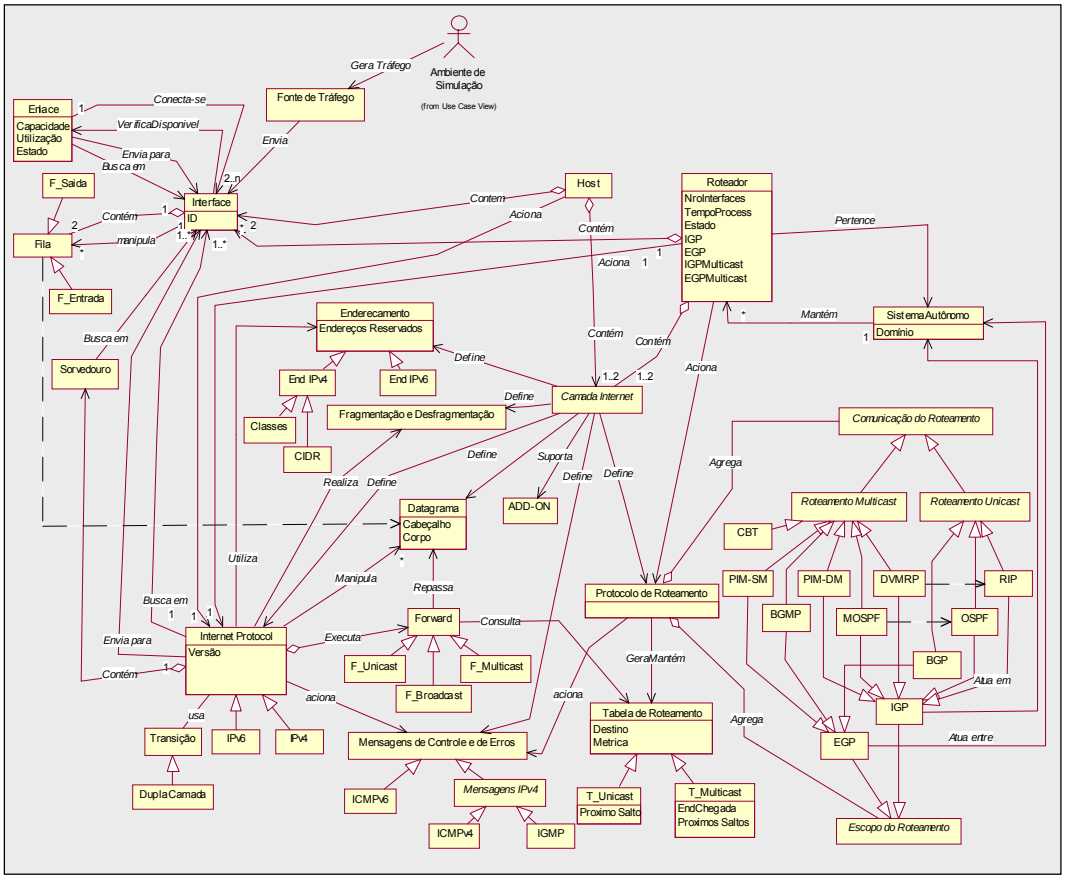


Figura 4.3: Diagrama de Conceitos Abrangente.

A necessidade de um diagrama mais simplificado, exigiu um estudo minucioso de abstração daqueles conceitos mais relevantes de forma reduzir a quantidade de conceitos e, consequentemente, a quantidade de interações entre eles. A figura 4.4 apresenta esse digrama o qual foi denominado "Diagrama de Conceitos Simplificado".

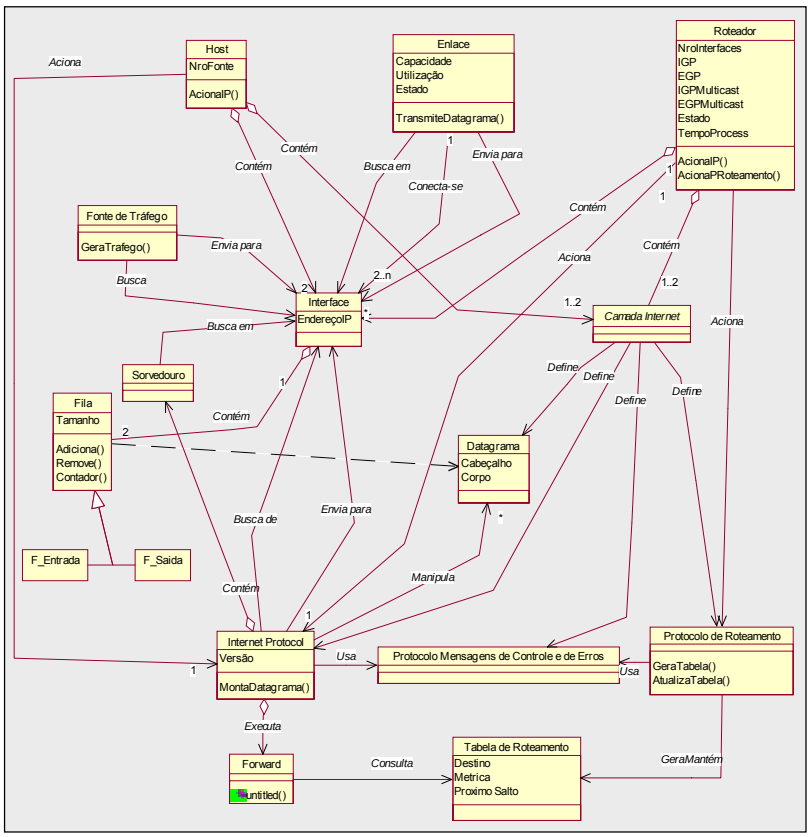


Figura 4.4: Diagrama de Conceitos Simplificado.

4.3.2. Diagramas de Seqüência

Para verificar o comportamento dinâmico do sistema (o que ele faz), sob o ponto de vista do usuário, a UML proporciona o artefato “diagrama de seqüência”. Um diagrama de seqüência possibilita que se descreva os eventos dos *use-cases*, e a ordem em que estes são gerados. Os atores é que provocam a ocorrência de eventos. É interessante que se gere diagramas de seqüência apenas para os *use-cases* mais importantes, e dentro destes, aqueles que representam as situações mais críticas [Larman, 98].

Apresentam-se nas figuras 4.5 e 4.6, respectivamente, os diagramas de seqüência relacionados com os *use-cases* “Receber e Repassar Datagramas” e “Criar e Manter Tabelas de Roteamento”. Esses *use-cases* foram escolhidos devido suas importâncias no contexto da especificação em questão.

No diagrama de seqüência da figura 4.5[7], o Ator “Ambiente de Simulação” invoca o componente (conceito) Fonte de Tráfego para gerar trafego. A fonte de tráfego envia o tráfego para a Interface do *Host* 1 (H1) o qual ele pertence. Percebendo a presença de uma mensagem

[7] Neste diagrama não estão representados explicitamente os componentes datagrama, IP, protocolo de roteamento, interface e suas respectivas filas de entrada e de saída da Interface, embora mencionado no texto para um melhor entendimento da descrição do sistema.

na fila de entrada de sua interface, com a aplicação, o IP de H1 busca a mensagem, monta o(s) datagrama(s) e envia cada datagrama para a fila de saída da Interface com o Roteador (R). O enlace entre H1 e R1 busca o datagrama na fila de saída da interface de H1 e envia para a fila de entrada da Interface de R1. Este, por intermédio de seu IP, percebe a presença de um datagrama na fila de entrada de uma de suas Interfaces e, então busca o datagrama, para realizar um processamento (*forward*) consultando a tabela de roteamento, para então enviar o datagrama para a fila de saída da Interface conforme indicação da tabela de roteamento.

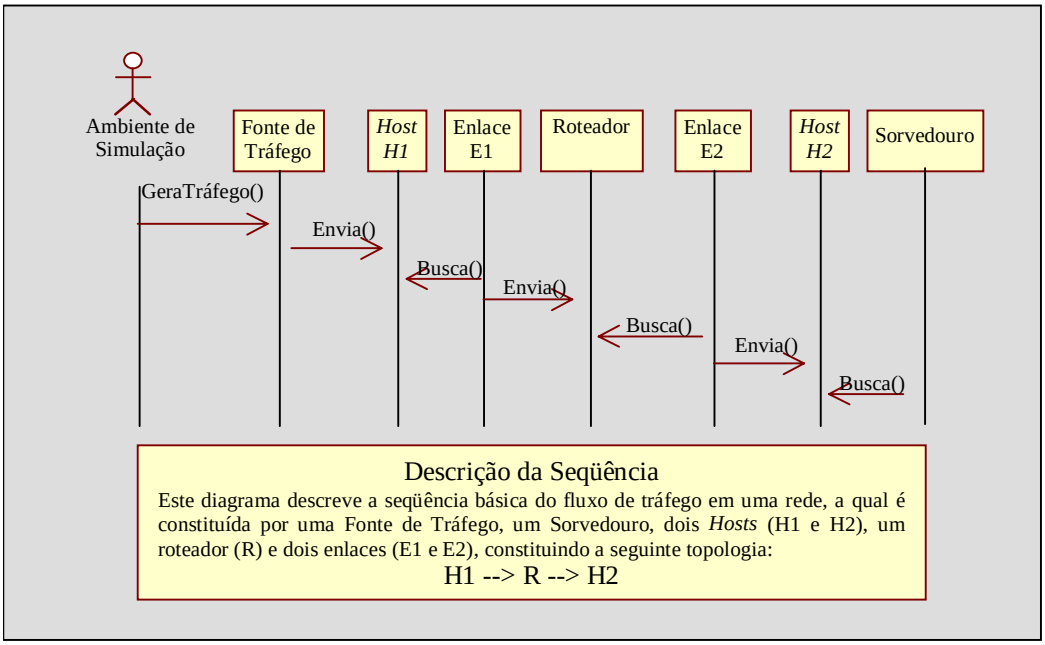


Figura 4.5: Diagrama de Seqüência: Receber e Repassar Datagramas.

O enlace entre R e H2 repete as operações feitas pelo enlace entre H1 e R enviando o datagrama para H2 que envia o datagrama para fila de saída da interface da aplicação, para que o sorvedouro busque e descarte o datagrama.

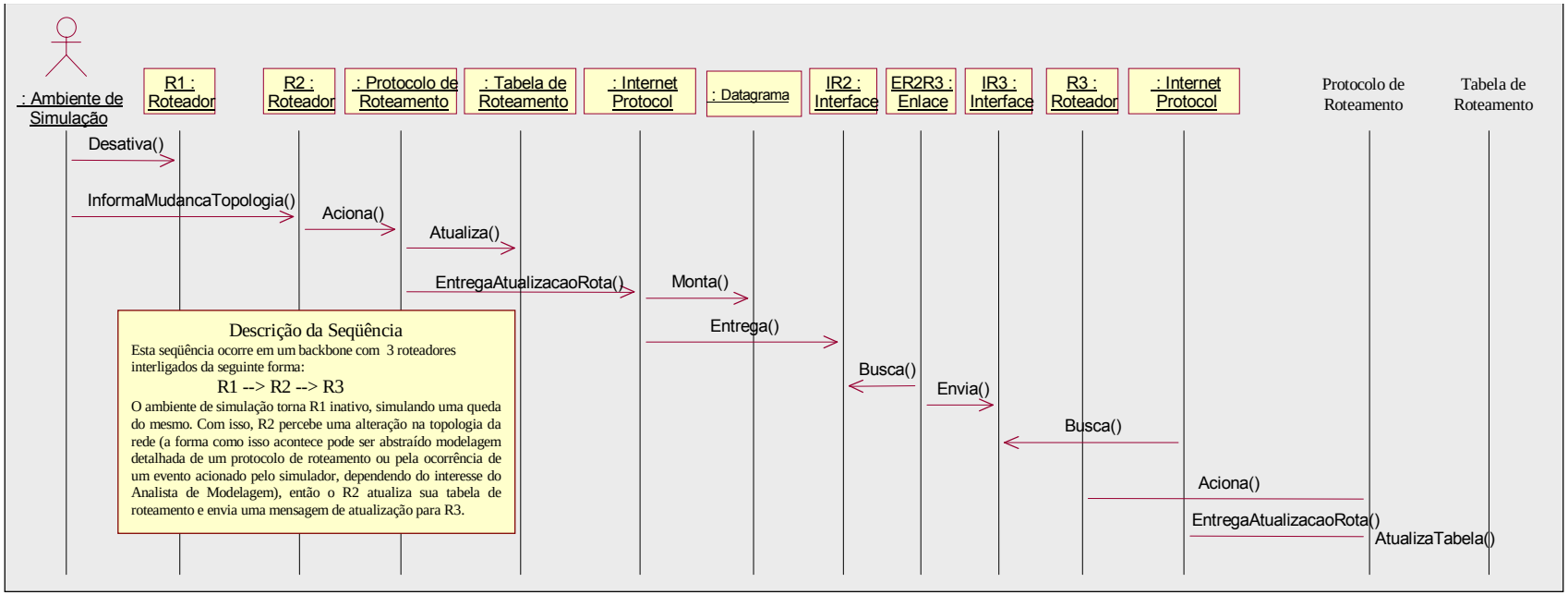


Figura 4.6:Diagrama de Seqüência: Manutenção da Tabela de Roteamento.

No diagrama de seqüência da figura 4.6, o Ator Ambiente de Simulação força a desativação do roteador1 (R1) e informa o roteador2 (R2) do fato ocorrido (a forma como o R2 é avisado da mudança da topologia da rede depende do nível de detalhamento da implementação dos protocolos de roteamento implementados pelo Analista de Modelagem, ou desenvolvedor do componente protocolo de roteamento). Então R2 aciona seu protocolo de roteamento, o qual atualiza sua tabela de roteamento, e então envia uma mensagem de atualização de rotas para a interface interna do IP de R2. O IP, por sua vez, monta um datagrama, contendo a mensagem de atualização de rotas, e envia o datagrama para sua interface (IR2). Então o enlace (entre R2 e R3) busca o datagrama em IR2 e envia para IR3. O IP do R3 busca o datagrama na interface de seu roteador, desmonta o datagrama e entrega para o protocolo de roteamento e, então atualiza a tabela de roteamento.

Dependendo do nível de detalhamento do protocolo de roteamento podem ser elaborados novos diagramas de seqüência.

4.3.3. Diagramas de Estados

Outro artefato, definido pela UML, designado a modelar o comportamento de dinâmico de um sistema é o "diagrama de estado", o qual ilustra os eventos e estados de um componente (conceito). Os diagramas de estados mostram o ciclo de vida de um objeto ou componente (conceito), através dos eventos a que está sujeito, suas transições (relacionamento entre dois estados) e os estados que ele tem entre tais eventos [Larman, 98].

A figura 4.7 apresenta o diagrama de estados do componente roteador. Este diagrama mostra que o estado inicial de um roteador é "Disponível" e de acordo com os eventos Chegada de Pacote, Saída de Pacote, Aciona IP e Queda na Rede ocorrem transições que o fazem mudar para um dos outros possíveis estados (Ocupado, Congestionado ou Inativo). A transição do estado Disponível para o estado Ocupado pode ocorrer com o acontecimento de dois eventos distintos (chegada de pacote ou Aciona IP), pois a execução do protocolo de roteamento independe da chegada de pacotes. Neste diagrama, também pode-se observar que para um roteador ser eliminado da rede (modelo) ele primeiro deve estar no estado Inativo.

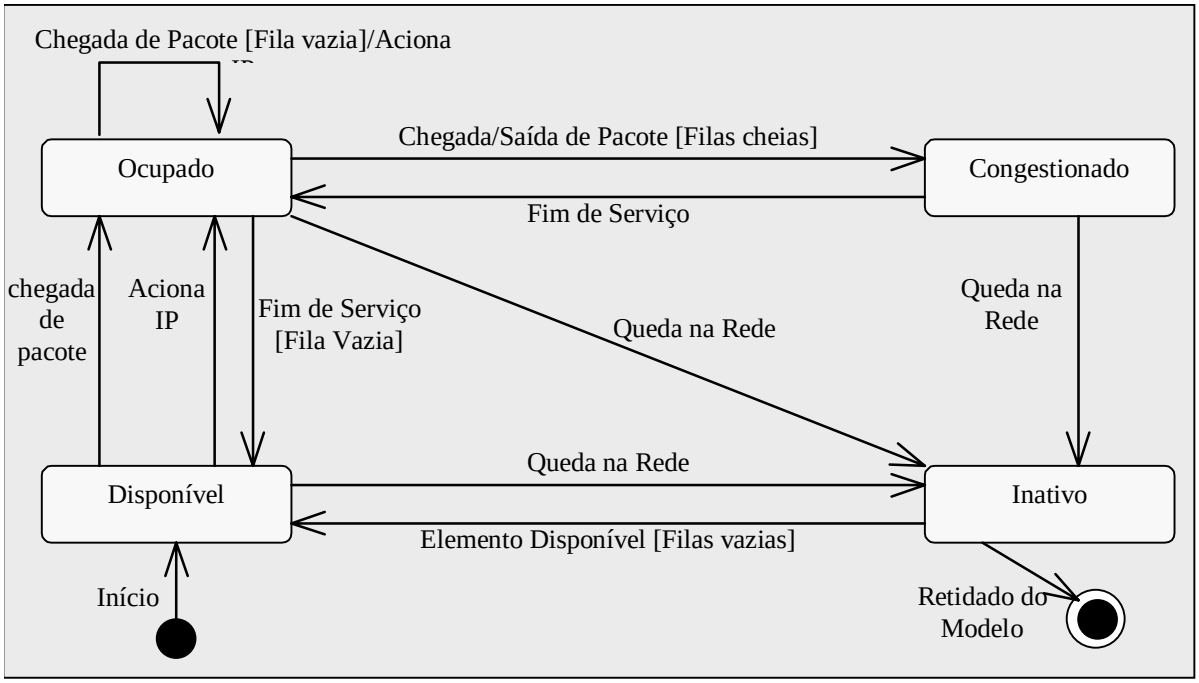


Figura 4.7:Diagrama de Estados: Componente (conceito) Roteador.

Outros diagramas de estados podem ser construídos de acordo com as necessidades de compreensão ou detalhamento do comportamento dinâmico do sistema, como por exemplo os diagramas de estado dos enlace e *host*, os quais estão sujeitos aos mesmos estados do componente roteador.

4.4. Fase de Projeto

A fase de projeto é uma extensão do modelo de análise visando sua implementação num computador. A ênfase do projeto é **como** fazer, cuja documentação resultante visa atender os anseios do desenvolvedor e não do o cliente. O projeto de um sistema pode ser dividido em duas partes [Larman, 98]: Projeto Arquitetural e Projeto Detalhado. O projeto detalhado somente é abordado no que condiz a identificação das propriedades e operações dos componentes.

4.4.1. Projeto Arquitetural

Em um projeto arquitetural é interessante decompor o sistema em camadas, definindo vários níveis de abstração. Análogo à decomposição de um sistema de informação, conforme apresentado em [Larman, 98], neste trabalho foi utilizado o esquema clássico de 3 camadas (figura 4.8):

- ✓ **Apresentação:** que trata da interface que permite o usuário configurar os componentes da rede TCP/IP. A parte superior da figura 4.8 apresenta um exemplo de protótipo para as telas de configuração dos componentes Fonte de Tráfego, Sorvedouro, *Host*, Roteador e Enlace. Esta definição busca contemplar os requisitos F7 e F8. Todas as entrada de dados apresentadas nos protótipos (ex.: Nome do Roteador, nº de Interfaces, etc.) são definidas como atributos dos componentes especificados. Os protótipos apresentados na figura 4.8 contemplam o requisito NF2.
- ✓ **Lógica da Aplicação**: esta camada é apresentada em duas sub-camadas:
  - **Componentes Alto Nível**: compreendem às unidades configuráveis em mais alto nível de abstração, correspondendo aos elementos necessários para modelar uma rede TCP/IP (Roteador, Enlace, Host, FonteTráfego e Sorvedouro. Esta definição busca contemplar o requisito F1;
  - **Componentes Baixo Nível**: são componentes utilizados pelos componentes de Alto Nível, em sua composição (camada Internet e Interface); Esta definição busca contemplar o requisito F2;
- ✓ **Armazenamento:** representa as estruturas dos dados existentes no contexto do domínio do problema. Essas estruturas estão organizadas em tabelas (de roteamento *unicast* e *multicast*), filas (de entrada e saída das Interfaces) e dados da configuração dos componentes do modelo. Entretanto, o tratamento destes dados não se inserem no escopo desta especificação. Tais decisões devem ser tomadas de acordo com o ambiente de simulação que acolherá os componentes aqui especificados.

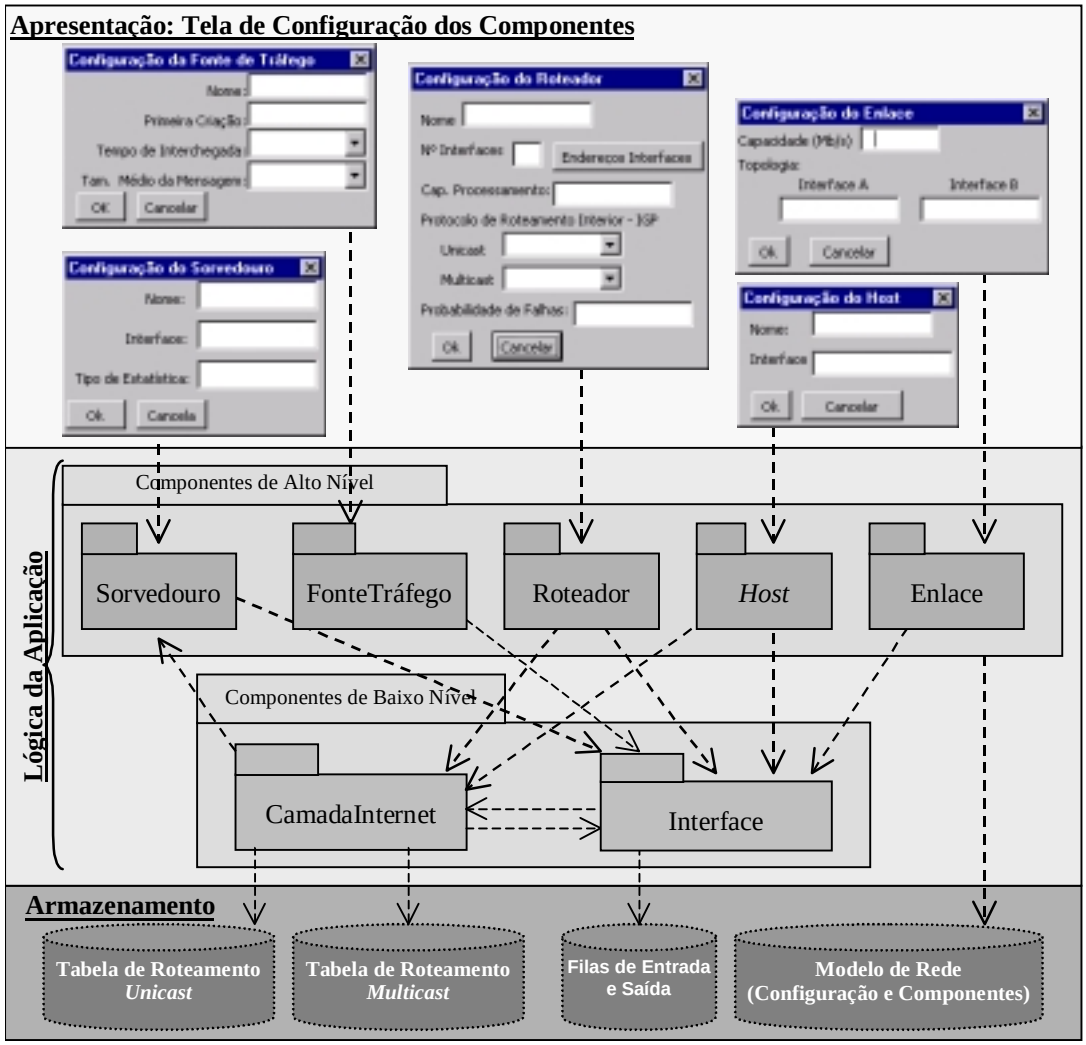


Figura 4.8: Modelo Arquitetural.

O escopo deste trabalho limita-se a especificar os componentes existentes na camada Lógica da Aplicação do modelo arquitetural (figura 4.8). Portanto, torna-se necessário especificar cada componente desta camada definindo seus devidos conceitos e interfaces. Na especificação deve estar claro quais são os serviços (operações) fornecidos e de quais componentes são requeridos serviços.

A UML não define nenhum artefato específico para representar componentes de software em alto nível. Este trabalho, para representar esses componentes adota o artefato “pacote” da notação UML. Um dos propósitos de se usar pacotes é tornar explícitas as dependências entre diferentes módulos do sistema em desenvolvimento. Essa arquitetura pode também apresentar as dependências entre os componentes da mesma camada, e entre os componentes de camadas distintas, como mostrado na figura 4.8. Contudo, para representar um componente

detalhadamente (sessão 4.4.2), com suas respectivas propriedades e serviços oferecidos (operações), foram adotadas os artefatos classe e <<interface>>[8] da notações UML.

O artefato <<interface>> da UML, o qual apresenta uma coleção de operações usadas para especificar um serviço de uma classe ou de um componente. <<Interfaces>> são utilizadas para visualizar, especificar, construir e documentar as "costuras" do sistema em questão. Sendo bem estruturada, uma <<interface>> provê uma clara separação entre a visão externa e interna de uma abstração, tornando possível compreender a mesma sem ter que mergulhar em detalhes de implementação [Rumbaugh, 99].

Como mostra a figura 4.9, os componentes Roteador e *Host* são compostos pelos componentes CamadaInternet e Interface. Pode-se observar que o componente CamadaInternet depende da <<interface>> provida pelo componente Interface e vice-versa. Portanto, antes de apresentar as especificações dos componentes de alto nível é interessante especificar os componentes de baixo nível que os compõe, como descritos na próxima sessão.

[8] No decorrer deste trabalho, ao reportar-se ao artefato "interface" da UML será utilizada a notação <<interface>> para evitar confusão com a denominação Interface que é um componente de Baixo Nível da camada Lógica da Aplicação, conforme mostrado na figura 4.8.

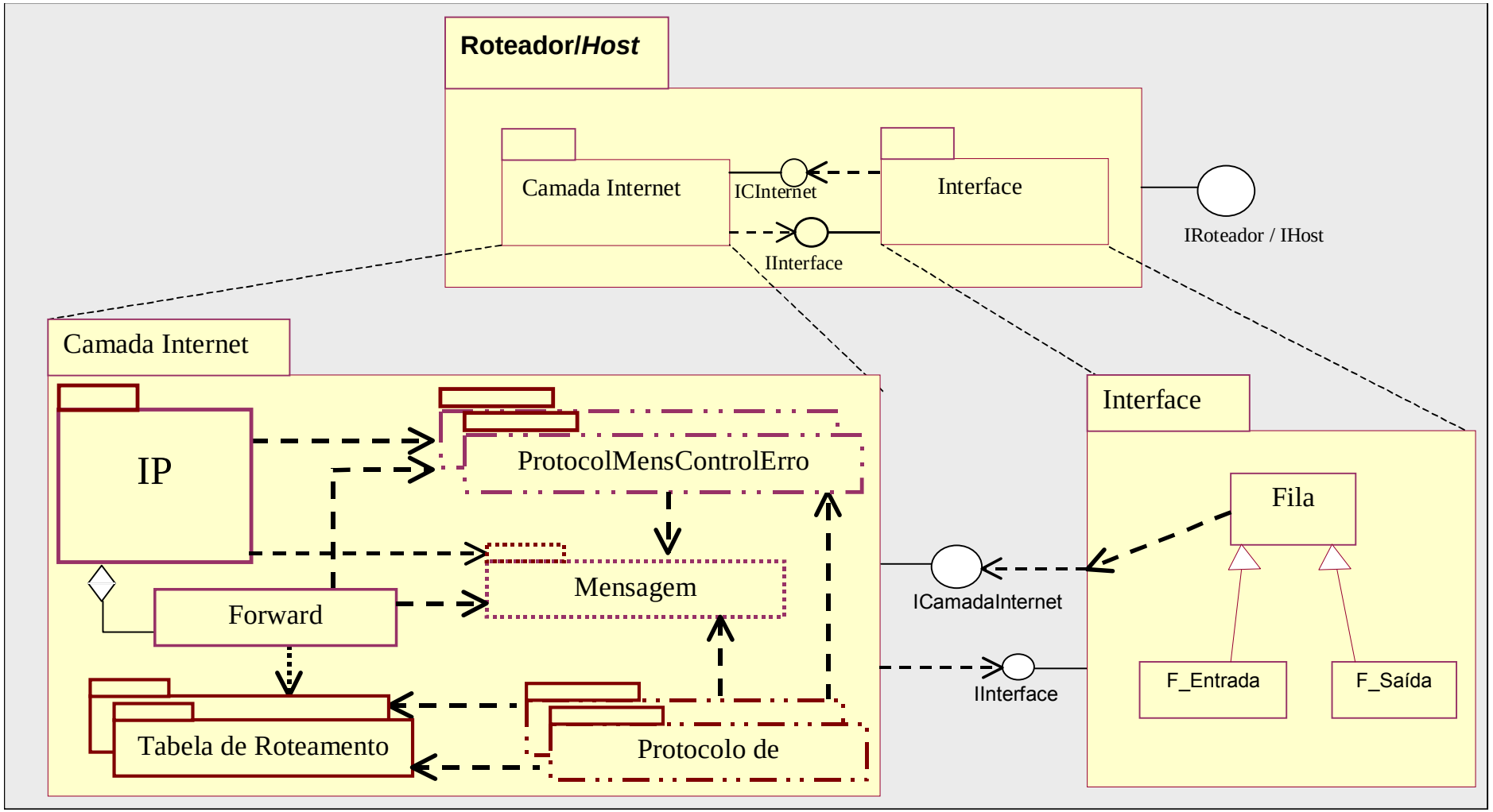


Figura 4.9: Composição dos Componentes Roteador e *Host*.

4.4.2. Projeto Detalhado

Esta sessão destina-se a identificar as propriedades e operações dos componetes identificados na sessão anterior sob forma de pacotes.

Devido ao componente *Interface* se relacionar diretamente com a maioria absoluta dos componentes, sua especificação será apresentada em primeiro lugar.

4.4.2.1. Interface

O componente Interface é um componente chave neste sistema, pois é através dele que todos os componentes de alto nível se comunicam. Estes relacionamentos estão documentados, tanto nas especificações de alto nível (sessão 3.1.Funcionalidades da Camada Internet), quanto nos diagramas de conceitos apresentados anteriormente (figuras 4.3 e 4.4). A figura 4.10 reforça a interação entre tais componentes, mostrando que o componente Interface contém uma fila de entrada e uma fila de saída (esta definição contempla o requisito **F11**). No entanto, os componentes que se utilizam deste é que buscam e enviam mensagens de e para suas filas. As unidades de informação de suas filas são definidas pelo componente *Mensagem*.

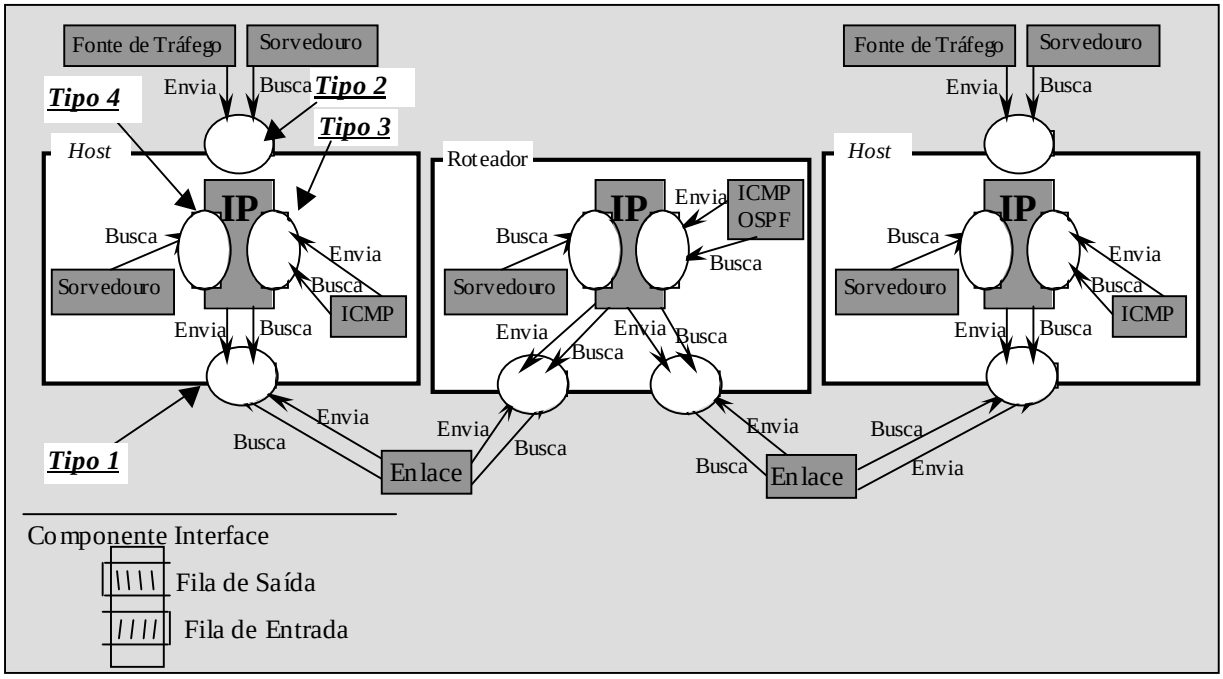


Figura 4.10:Comunicação de uma Rede TCP/IP

Desta forma, conforme a figura 4.10, o componente Interface pode ser utilizado em quatro tipos de Interfaces, a saber:

- ✓ **Tipo 1: Interface com a Camada Inferior**: a qual representa a Interface com a camada "Interface de Rede" da arquitetura TCP/IP. Estas Interfaces estão presentes

tanto no componente *Host* quanto no componente Roteador, as quais, são acopladas nas extremidades do componente Enlace, para ligar dois nós da rede (*Host* com Roteador, ou Roteador com Roteador);

- ✓ **Tipo 2***: **Interface* **com a Camada Superior** : está presente apenas no componente *Host*, e está acoplada ao componente *FonteTráfego* (para receber as *Mensagem* pela fila de entrada) e ao componente *Sorvedouro* (o qual retira as *Mensagens* da fila de saída);
- ✓ **Tipo 3***: **Interface* **entre Protocolos da Camada Internet**: esta I*nterface* está presente no componente *IP*. E por intermédio dela é que o componente *IP* viabiliza a entrada/saída de *Mensagens* de controle na rede, ao acopla-la nos componentes ProtocolMensControlErro e ProtocoloRoteamento;
- ✓ **Tipo 4***: **Interface* **para descarte de pacotes**: esta *Interface* está presente no componente *IP* e tem a função de receber, em sua fila de saída, os *datagramas* a serem descartados por este protocolo. Esta *Interface* está acoplada a um componente *Sorvedouro* usado especificamente para isto. Esta *interface* é definida estritamente por esta especificação, por este motivo não foi definida na figura 3.3 do capítulo anterior.

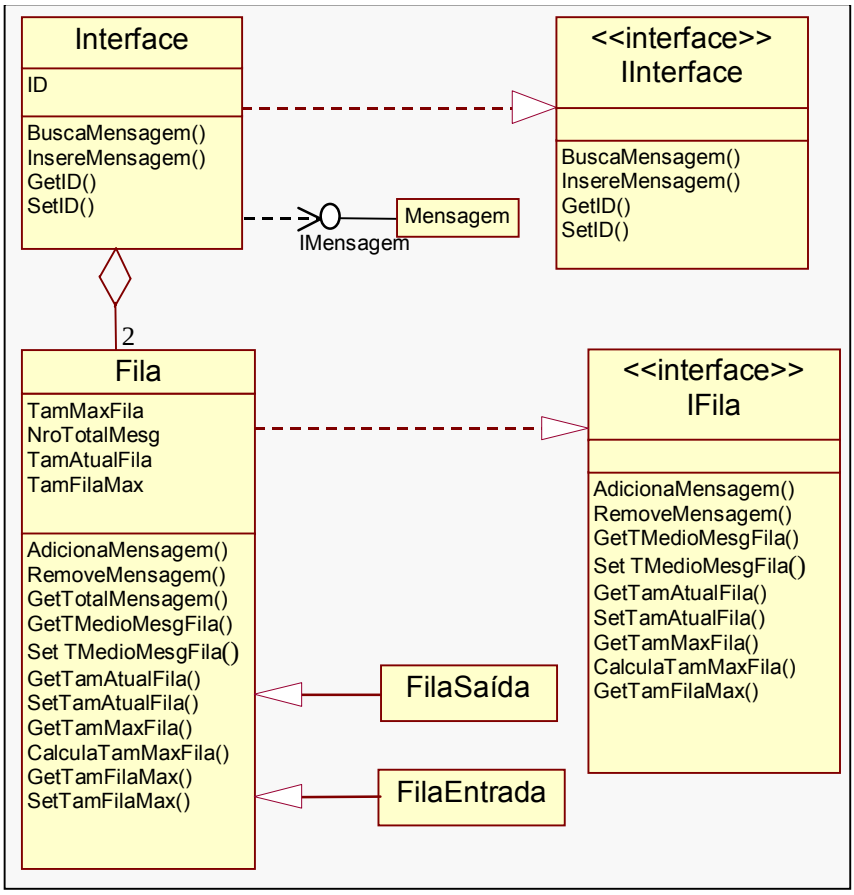


Figura 4.11: Componente Interface.

A figura 4.11 apresenta o componente *Interface* com suas devidas propriedades e operações, juntamente com a realização de sua <<interface>> (IInterface) e suas dependências com relação aos serviços oferecidos por outros componentes. A seguir apresentam-se as descrições das propriedades e operações do componente *Interface*:

- ✓ **ID:** identificador da Interface. Não existem dois componentes Interface com o mesmo identificador. Permite operações de *Get* e *Set;*
- ✓ **BuscaMensagem**(): operação do componente Interface que busca uma Mensagem em uma Fila;
- ✓ **InsereMensagem**(): operação que insere uma Mensagem em uma Fila;
- ✓ Segue as propriedades e operações do componente Fila:
- ✓ **TamMaxFila**: número máximo de Mensagem que uma Fila pode armazenar. Permite operações de *Get* e *Set;*
- ✓ **NroTotalMesg**: armazena número total de Mensagens que passaram pela Fila no decorrer da simulação. Permite operações de *Get* e *Set;*
- ✓ **TamAtualFila**: armazena o número total de Mensagens, na fila, em um determinado instante. Permite operações de *Get* e *Set;*
- ✓ **TamFilaMax**: medida de desempenho que armazena o tamanho máximo que uma fila alcançou durante a simulação. Permite operações de *Get* e *Set;*
- ✓ **AdicionaMensagem**(): operação que coloca uma Mensagem na fila;
- ✓ **RemoveMensagem**(): operação que remove uma Mensagem na fila;
- ✓ **GetTotalMensagem**() operação que faz o somatório do número de mensagens em uma fila;
- ✓ **GetTMedioMesg**(): operação que retorna o Tamanho Médio das Mensagens que chegam em uma Fila. Permite operação *Set;*

4.4.2.2. CamadaInternet

Conceitualmente a Camada Internet define várias funcionalidades (implementadas por protocolos), conforme a figura 4.12, que devem interagir para prover a interconexão de redes de computadores, na arquitetura TCP/IP. No entanto, existem mais de um protocolo, de diversas versões, que podem implementar a mesma funcionalidade, como é o caso dos Protocolos de Roteamento Dinâmico. Desta forma, percebe-se a necessidade de se especificar um modelo para cada protocolo de roteamento que se deseja avaliar.

No caso deste trabalho, em que se busca a realização de uma especificação reutilizável de componentes para avaliação de redes TCP/IP, é interessante que se defina o componente CamadaInternet como sendo um conjunto de componentes prontos e "semiprontos" devidamente relacionados, caracterizando um *Framework* de Componentes. Este componente define cinco novos componentes (IP, Forward, ProtocoloRoteamento, TabelaRoteamento,

ProtocolMensControlErro), cujas dependências estão representadas na figura 4.12. Os pacotes cujas bordas estão tracejadas representam componentes semiprontos, aos quais devem ser agregadas as funcionalidades (ou protocolos) desejadas pelo Analista de Modelagem. O componente ProtocoloRoteamento é um componente genérico que necessita a agregação de um protocolo de roteamento *unicast* (OSPF) e um protocolo de roteamento *multicast* (ex. MOSPF). No caso do componente TabelaRoteamento, a figura mostra o pacote duplicado com o intuito de ressaltar a possibilidade da existência de duas tabelas de roteamento (*unicast* e *multicast*). O componente mensagem é considerado incompleto por esta especificação não definir os campos de seu cabeçalho, alem de existirem 4 tipos de mensagens (cada qual sendo usada em suas devidas situações), como abordado mais adiante. O componente ProtocolMensControlErro é considerado semipronto por permitir que o usuário da especificação defina quais são as mensagens específicas que ele deve tratar.

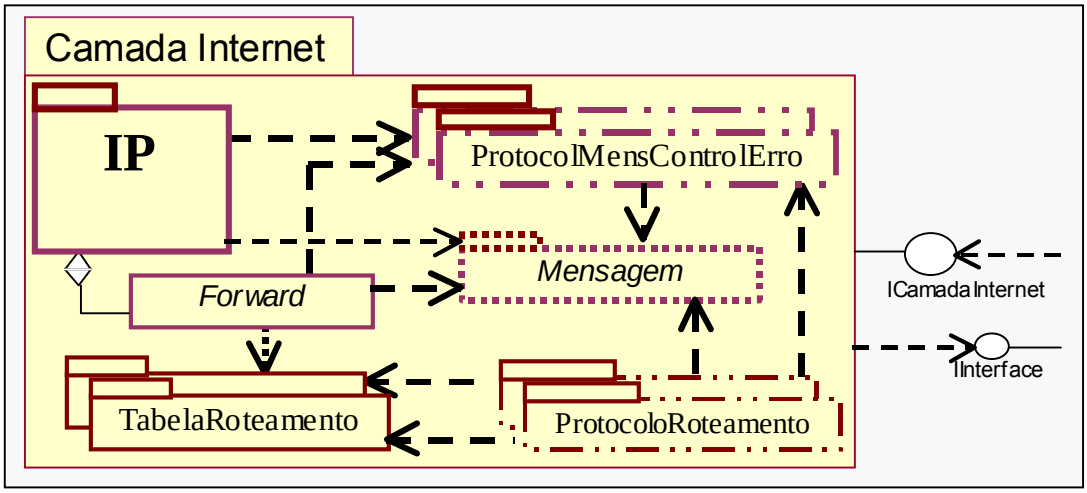


Figura 4.12: *Framework* de Componentes da Camada Internet.

O *framework* de componentes CamadaInternet, como mostra a figura 4.12, agrega todos os componentes que implementam as funcionalidades pertinentes a esta camada da arquitetura TCP/IP. Os componente Roteador e Host utilizam-se de 1 (uma) instância deste componente para ter acesso aos protocolos que vão implementar. Estas definições buscam contemplar aos requisitos **F8** e **NF1**.

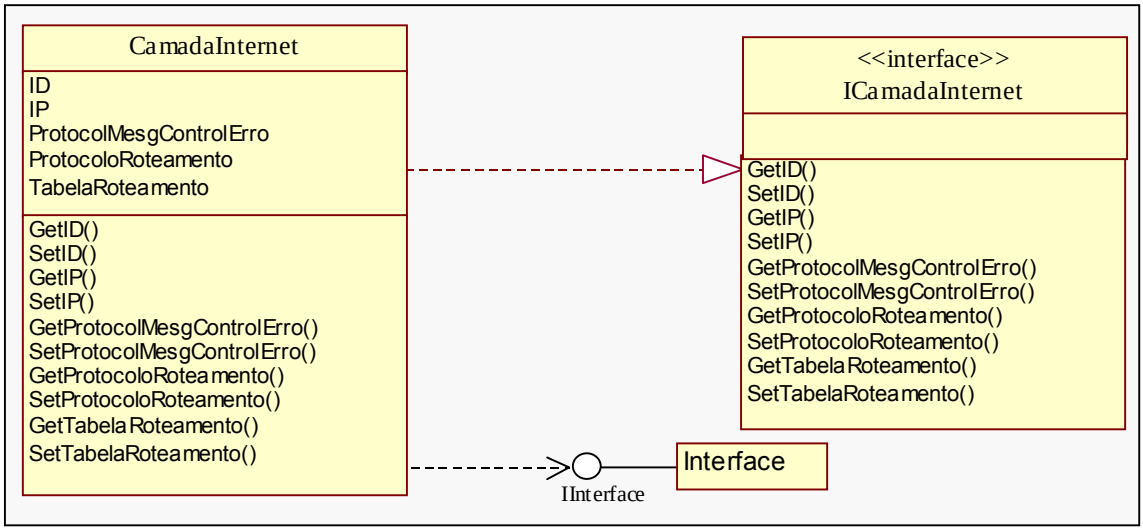


Figura 4.13: Componente CamadaInternet.

A figura 4.13 apresenta o componente CamadaInternet com suas devidas propriedades e operações, juntamente com a realização de sua <<interface>> (ICamadaInternet) e suas dependências com relação aos serviços oferecidos por outros componentes. A seguir apresentam-se as descrições das propriedades e operações do componente CamadaInternet:

- ✓ *ID*: identifica o componente *CamadaInternet* como parte integrante de um componente *Roteador* ou *Host*. Permite operações Get e Set;
- ✓ Cada protocolo agregado a este componente (*ProtocoloRoteamento*, *ProtocolMensControlErro* e *IP*) e o componente *TabelaRoteamento.* Permite operações Get e Set;

As operações e propriedades de cada um destes componente serão detalhadas a seguir, tais componentes contemplam o requisito **F4**.

4.4.2.3. IP e Forward

O componente IP (figura 4.14) é o componente que efetiva realmente o repasse de datagramas. Para tal ele agrega o componente Forward que, de acordo com as configurações realizadas pelo usuário (nos Roteadores), implementa as funcionalidades de comunicação *unicast* e *multicast*. O componente IP agrega um componente Interface específico para se comunicar com os componentes ProtocolMesgControlErro e ProtocoloRoteamento e outro para viabilizar o descarte de datagramas.

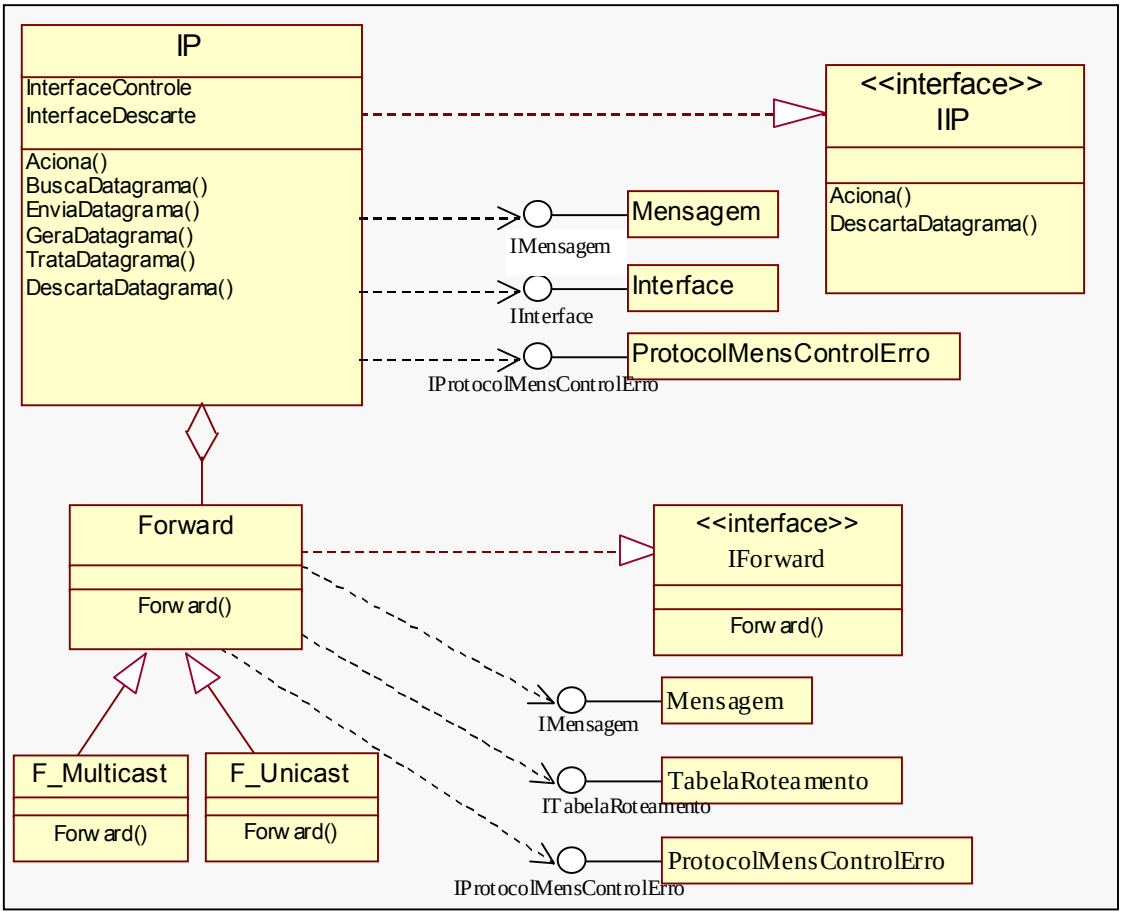


Figura 4.14: Componente IP.

A figura 4.14 apresenta o componente IP com suas devidas propriedades e operações, juntamente com a realização de sua <<interface>> (IIP) e suas dependências com relação aos serviços oferecidos por outros componentes. A seguir apresentam-se as descrições das propriedades e operações do componente IP:

- ✓ *InterfaceControle*: está propriedade representa a agregação de um componente *Interface* para permitir que o componente *IP* receba mensagens dos componentes *ProtocolMensControlErro* e *ProtocoloRoteamento*;
- ✓ *InterfaceDescarte*: está propriedade representa a agregação de um componente *Interface* para permitir que o componente *IP* descarte pacotes utilizando a operação *DescartaDatagrama*(), isto implica no acoplamento de um componente *Sorvedouro*.
- ✓ *Aciona*(): esta operação permite que os componentes que o contém (*Host* e *Roteador*) iniciam a execução do componente *IP*;
- ✓ *BuscaDatagrama*() e *EnviaDatagrama*(): estas operações são executadas pelo *IP* para buscar/enviar mensagens em/para o componente *Interface*; Esta definição é parte integrante da contemplação do requisito F4.

- ✓ *GeraDatagrama*(): quando o componente *IP* recebe uma mensagem do componente *Interface* ligada a ao componente *FonteTráfego* ou à *InterfaceControl*, ele gera um cabeçalho e coloca a mensagem recebida no corpo deste *datagrama*. Esta definição visa contemplar o requisito F9;
- ✓ *DescartaDatagrama*(): esta operação utiliza-se da interface com o componente *Sorvedouro* para eliminar um *datagrama* da rede.
- ✓ *Forward*(): esta operação está presente no componente *Forward*, o qual está agregado ao componente *IP*. Caracteriza-se aqui, a existência de polimorfismo na implementação dos algoritmos de das comunicação *unicast* e *multicast*; Esta operação também consulta o componente *TabelaRoteamento*, Esta operação (componente) é parte integrante da contemplação do requisito F4 e F6.

4.4.2.4. Protocolo Mensagens de Controle e Erros

O componente ProtocolMensControlErro, apresentado na figura 4.15, define a <<interface>> IProtocolMensControlErro para que outros componentes possam acioná-lo. Este componente necessita apenas dos serviços do componente Mensagem. Dependendo do nível de detalhamento desejado pelo usuário deve ser permitida a extensão deste componente para agregação das funcionalidades dos protocolos específicos ICMP e IGMP. Este componente visa contemplar o requisito **F5**.

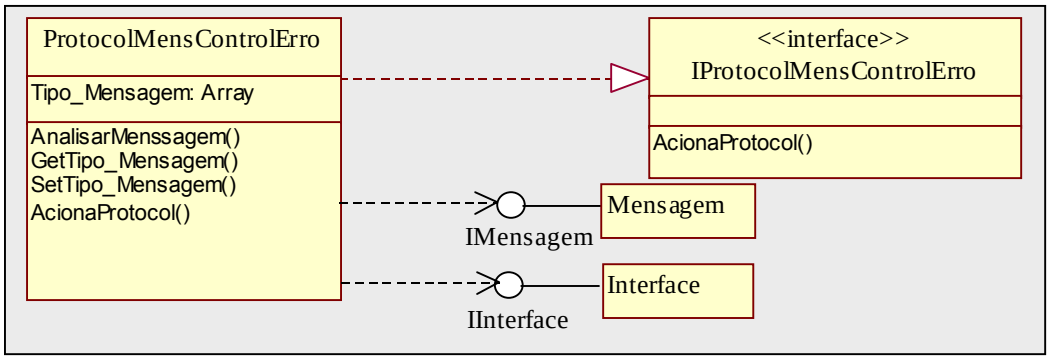


Figura 4.15. Componente Protocolo Mensagem de Controle e Erros.

- ✓ *Tipo_Mensagem*: esta propriedade apresenta todos os tipos de mensagens de controle e de erros implementadas pelo protocolo no modelo;
- ✓ *AnalisarMensagem*(): esta operação, ao receber uma requisição de uso (*AcionaProtocolo*) de outro componente (*IP* ou *ProtocoloRoteamento*) deve selecionar qual o tipo de mensagem de controle ou de erro correspondente para montar sua *Mensagem*. Esta operação utiliza-se das operações Get e Set de *Tipo_Mensagem*.

4.4.2.5. Protocolo de Roteamento

Este componente tem a função de gerar e manter as entradas do componente TabelaRoteamento. Um componente CamadaInternet pode Ter duas instâncias deste componente (*multicast* e *unicast*). Esta definição busca contemplar parte do requisito **F4.** Por se tratar de um componente semipronto este componente visa também contemplar o requisito **NF1**.

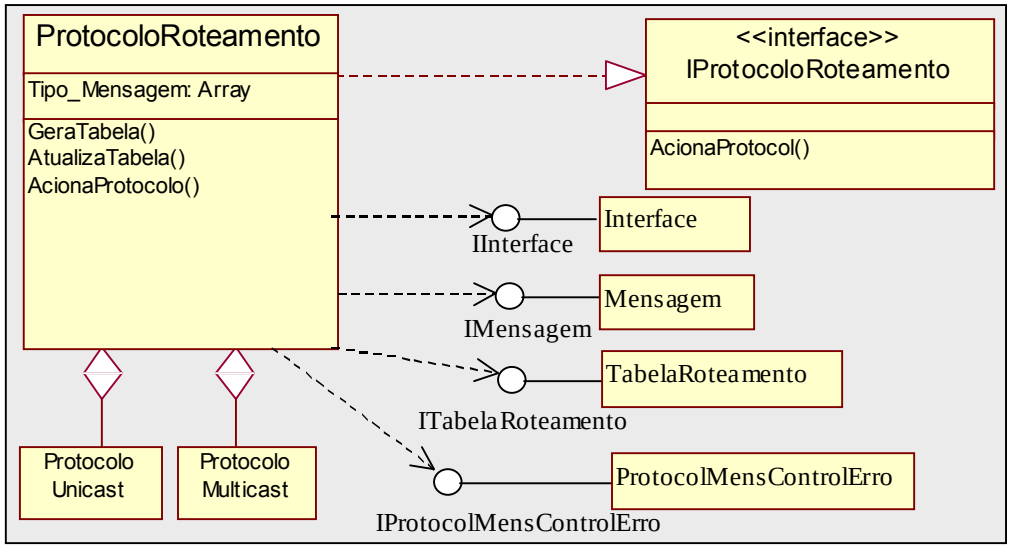


Figura 4.16: Componente Protocolo de Roteamento.

A figura 4.16 apresenta o componente "semipronto" ProtocoloRoteamento o qual deve agregar um protocolo de roteamento *unicast* (ex. RIP, OSPF) e um protocolo *multicast* (ex. DVMRP, MOSPF). Os protocolos que atuarão em um modelo específico serão selecionados pelo usuário a partir de uma interface de configuração do roteador oferecida pelo Ambiente de Simulação. Não está no escopo deste trabalho especificar tais protocolos, entretanto é definido que estes protocolos devem implementar as operações básicas de um protocolo de roteamento (GeraTabela(), AtualizaTabela(), AcionaProtocolo()). A operação AcionaProtocolo(), disponibilizada na interface IProtocoloRoteamento, é utilizada pelo roteador para colocar o protocolo de roteamento em ação, no início da simulação, permitindo que o *host* possa utilizar o componente CamadaInternet sem rodar nenhum protocolo de roteamento.

4.4.2.6. Tabela de Roteamento

A TabelaRoteamento (figura 4.17) oferece a interface ITabelaRoteamento que permite a consulta e manutenção em suas entradas. Este componente não requer serviços de outros componentes. A mesma é gerada e mantida pelo componente ProtocoloRoteamento e consultada pelo componente Forward***. A*** TabelaRoteaemento *agrega o componente chamado Rota, o qual é uma generalização dos componentes* RotaUnicast *e* RotaMulticast*, os quais armazenam as entradas na tabela (Endereço de destino, Métrica e Interface de Saída). Este componente também tem participação no contemplação do requisito F4.*

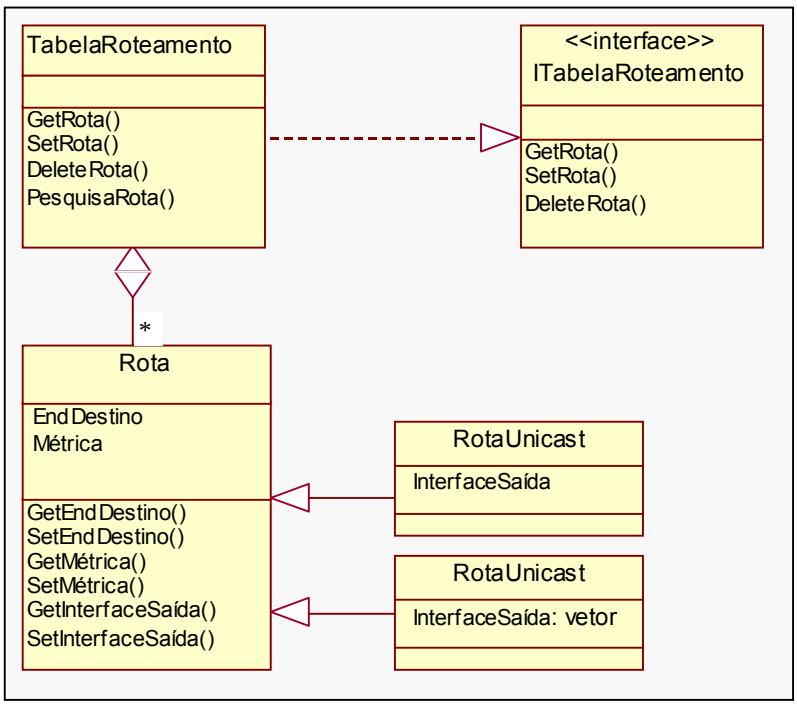


Figura 4.17: Componente Tabela de Roteamento.

A figura 4.17 apresenta o componente Mensagem com suas devidas propriedades e operações, juntamente com a realização de sua <<interface>> (ITabelaRoteamento) e suas dependências com relação aos serviços oferecidos por outros componentes. A seguir apresentam-se as descrições de suas propriedades e operações:

- ✓ GetRota(), SetRota() e DeleteRota(): são operações do componente TabelaRoteamento responsáveis pela manutenção (Lê, Escreve e deleta respectivamente) do componente TabelaRoteamento;
- ✓ PesquisaRota(): operação que procura uma entrada no componente TabelaRoteamento;
- ✓ O componente Rota define o conteúdo do componente TabelaRoteamento e contém os seguintes campos:
- ✓ EndDestino: Campo da rota que contém o endereço destino;
- ✓ Métrica: representa o custo da rota em questão;
- ✓ InterfaceSaída: identificador do componente Interface que o datagrama destinado a EndDestino deve seguir. No componente RotaMulticast podem existir vários identificadores de Interfaces.

4.4.2.7. Mensagem

O componente Mensagem (figura 4.18) define as unidades de informação utilizadas na rede aqui especificada. O datagrama é a unidade de informação que trafega em uma rede TCP/IP. O Stream é o nome dado a informação proveniente do componente fonte de tráfego,

representando dados de uma aplicação. MensagemRoteamento são mensagens utilizadas pelos protocolos de roteamento, portanto seu formato varia de acordo com o protocolo a ser utilizado (podem existir mais de um tipo de MensagemRoteamento). As MensagemControle são usadas pelo protocolo de Mensagens de Controle e Erros. Todos os 3 últimos tipos de mensagens apresentadas são encapsuladas pelo IP, em um datagrama, para serem repassadas pela rede. Este componente não necessita utiliza nenhum serviço externo a ele próprio. Pode ser estudado o caso da especificação deste componente fora do *Framework* de Componente CamadaInternet. Optou-se por defini-lo como parte integrante deste *Framework* de Componentes por questões conceituais. Acredita-se que a nível de implementação se perceberá que alternativa se enquadra melhor.

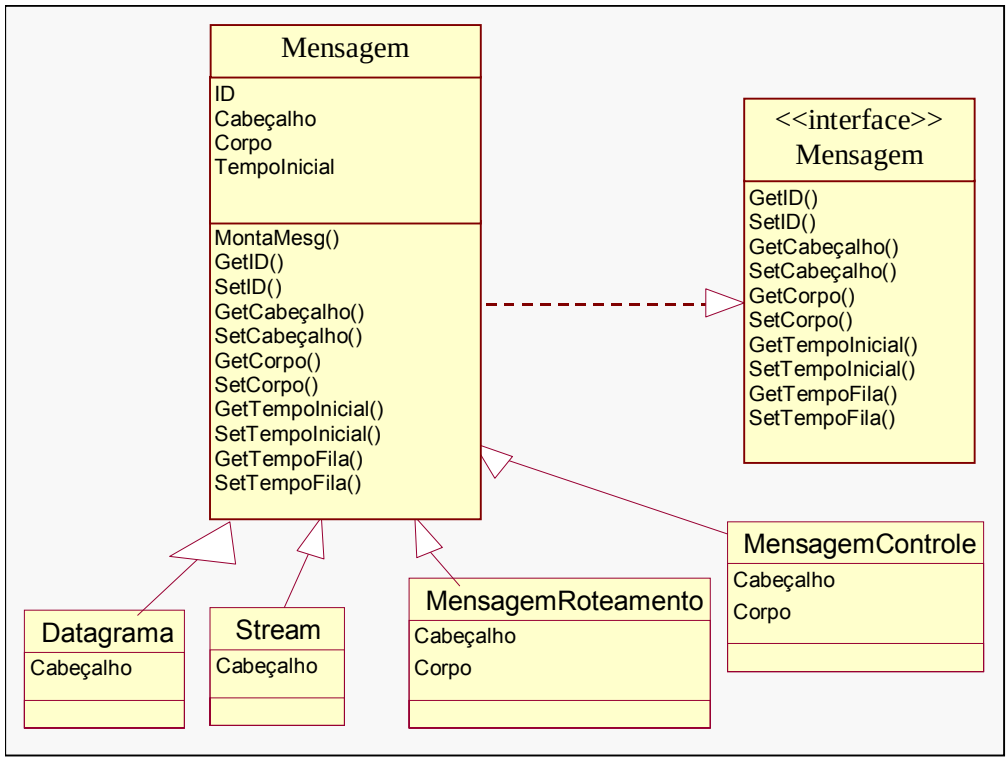


Figura 4.18: Componente Mensagem.

A figura 4.18 apresenta o componente Mensagem com suas devidas propriedades e operações, juntamente com a realização de sua <<interface>> (IMensagem) e suas dependências com relação aos serviços oferecidos por outros componentes. A seguir apresentam-se as descrições de suas propriedades e operações:

- ✓ *ID*: identificador do componente Mensagem. Cada instancia deve ser único;
- ✓ *Cabeçalho*: contém as informações pertinentes a unidade de informação em si, tal como endereço destino, tipo de protocolo que o gerou, tamanho. O conteúdo exato do cabeçalho depende dos interesses do Analista de Simulação. O cabeçalho do datagrama (tanto IPv4 quanto IPv6) é encontrado, na integra, no capítulo 2.

- ✓ *Corpo*: o conteúdo da informação em si, este campo, em determinadas situações de simulação pode ser vazio;
- ✓ *TempoInicial*: registra o momento em que a mensagem foi gerada.
- ✓ *MontaMesg*(): operação que monta uma mensagem, atribuindo os devidos valores do cabeçalho e corpo da mensagem.

4.4.2.8. Host

O Componente Host representa um elemento que conecta as aplicações do usuário (FonteTráfego) a uma rede. Cada Host é composto por um componente CamadaInternet e dois componentes Interface (cada Interface contém uma fila de entrada e uma fila de saída), definicção esta contempla o requisito **F10**. Um componente Interface conecta o Host a um componente Roteador, por intermédio de um componente Enlace. O segundo componente Interface conecta o componente Host (do ponto de vista da CamadaInternet) a aplicações do usuário, onde a fila de entrada(*host*, de origem) recebe fluxo de dados de um ou vários componentes FonteTráfego e a fila de saída está relacionada a entrega de dados a um Sorvedouro. O componente Host utiliza os serviços oferecido pelos componentes Interface e CamadaInternet. O componente Host deve ser configurado pelo usuário.

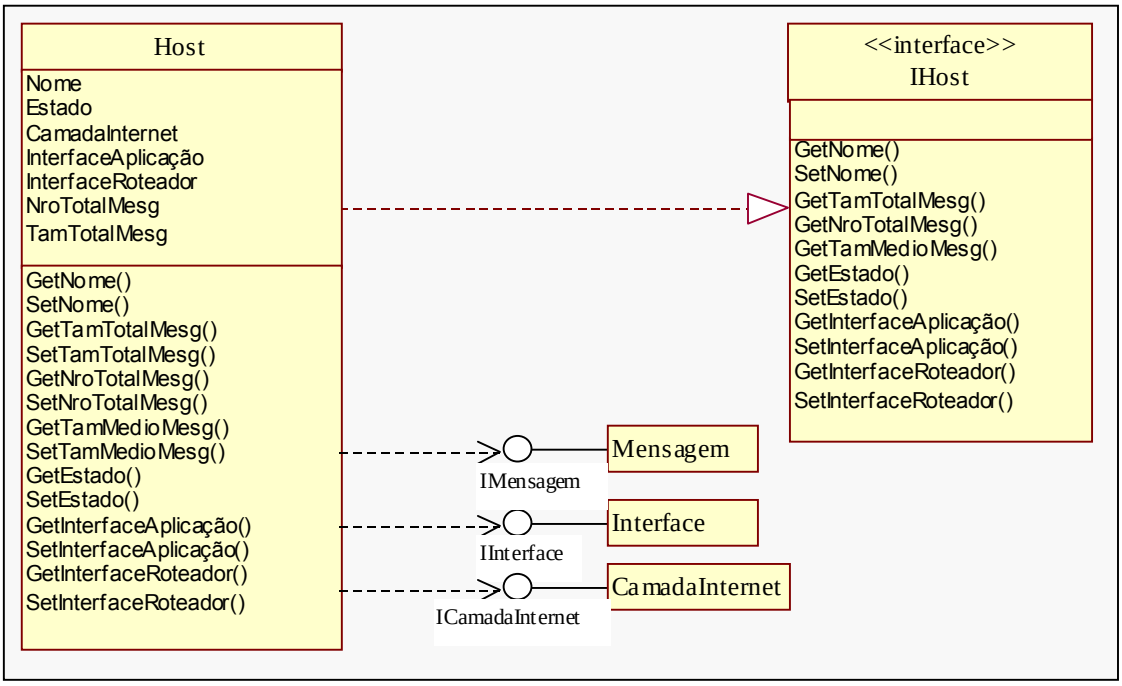


Figura 4.19: Componente *Host*.

A figura 4.19 apresenta o componente Host com suas devidas propriedades e operações, juntamente com a realização de sua <<interface>> (IHost) e suas dependências com relação aos serviços oferecidos por outros componentes. A seguir apresentam-se as descrições de suas propriedades e operações:

- ✓ *Nome: identifica o componente Host.* Permite operações Get e Set;*;*
- ✓ *Estado: indica se o componente está ativo (ocupado ou disponível) ou inativo;* Permite operações Get e Set;
- ✓ *CamadaInternet: identifica um componente CamadaInternet, o qual pertencente ao componente Host;*
- ✓ *InterfaceAplicação: identifica um componente Interface, o qual pertence ao componente Host. Este componente define a comunicação do componente CamadaInternet do Host com uma ou mais aplicações, as quais são representadas pelos componentes FonteTráfego e Sorvedouro.*
- ✓ *InterfaceRoteador: identifica um componente Interface, o qual viabiliza a comunicação do componente Host com o componente Roteador, por intermédio de um componente Enlace;*
- ✓ *NroTotalMesg: Medida de desempenho que armazena o número total de mensagens tratadas pelo componente Host;* Permite operações Get e Set;*, entretanto disponibiliza em sua <<interface>> apenas o Get;*
- ✓ *TamTotalMesg: Medida de desempenho que armazena o somatório dos tamanho de todas as mensagens tratadas pelo componente Host;* Permite operações Get e Set; *entretanto disponibiliza em sua <<interface>> apenas o Get;*
- ✓ *SetTamMedioMesg(): calcula o tamanho médio das mensagens recebidas pelo componente Host; Esta informação é calculada a partir da divisão de TamTotalMesg por NroTotalMesg;*
- ✓ *GetTamMedioMesg(): disponibiliza, através da <<interface>> do componente Host o resultado obtido em SetTamMedioMesg();*

4.4.2.9. Fonte de Tráfego

O componente FonteTráfego gera tráfego de dados que representam uma aplicação específica. Várias fontes de tráfego podem ser conectadas ao componente Interface de um componente Host, por exemplo, para modelar uma transmissão multimídia. Esta proposição contempla o requisito **F12**, utilizando-se das propriedades TempoInterchegada, NroMaxStream e TamanhoStream. Este componente deve ser configurado pelo usuário.

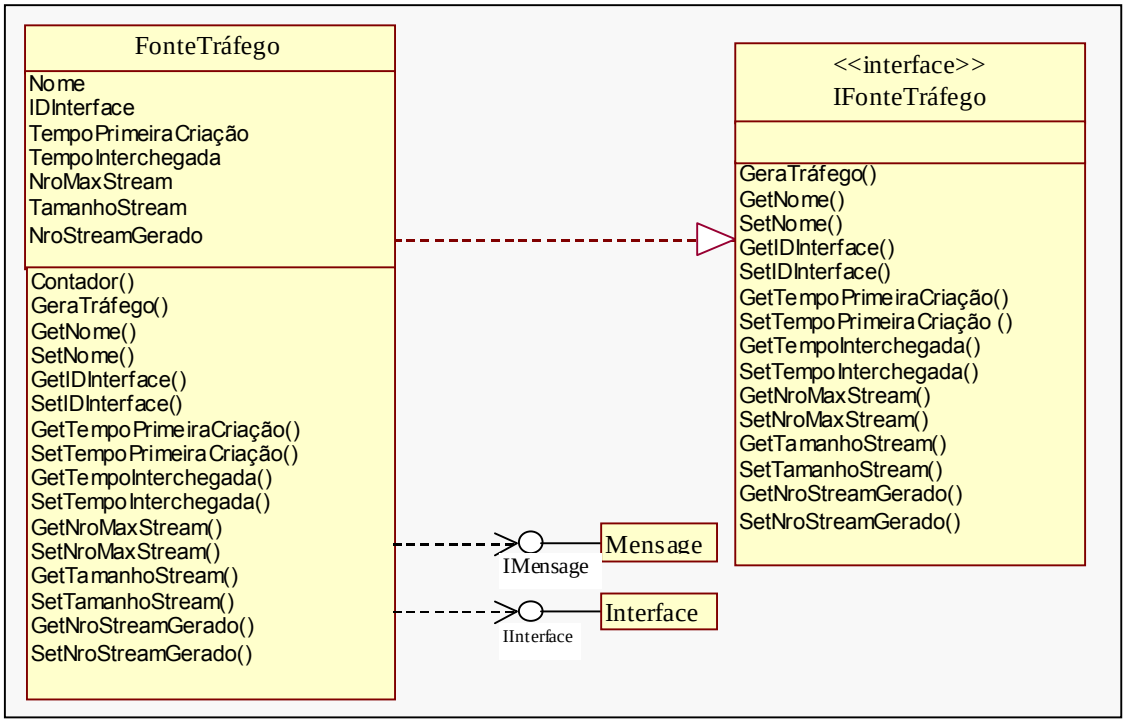


Figura 4.20: Componente FonteTráfego.

A figura 4.20 apresenta o componente FonteTráfego com suas devidas propriedades e operações, juntamente com a realização de sua interface (IFonteTráfego) e suas dependências com relação aos serviços oferecidos por outros componentes. A seguir apresentam-se as descrições de suas propriedades e operações:

- ✓ *Nome: Identifica o componente FonteTráfego.* Permite operações Get e Set;
- ✓ *IDInterface*: identifica o componente *Interface* pertencente ao componente *Host* pela qual ele está conectado; Permite operações Get e Set;
- ✓ *TempoPrimeiraCriacao*: indica o momento em que se inicia a geração de tráfego, durante a simulação. Permite operações Get e Set;
- ✓ *TempoInterchegada*: pode ser uma constante ou uma função (ex. Função de Distribuição de Probabilidade) que define o tempo entre a geração de duas unidades de informação. Permite operações Get e Set;
- ✓ *NroMaxStream*: número máximo de unidades de informações que podem ser gerado. Permite operações Get e Set;
- ✓ *TamanhoStream*: pode ser uma constante ou uma função (ex. Função de Distribuição de Probabilidade) que define o tamanho das unidades de informação a serem geradas. Permite operações Get e Set;
- ✓ *GeraTráfego*(): operação que gera as unidades de informações;

4.4.2.10. Sorvedouro

Este componente corresponde a uma aplicação destino, ou uma função de descarte de datagramas do componente IP, o qual está ligado ao componente Interface pertencente a um componente Host ou ao componente IP respectivamente. Este componente e tem a função de buscar os datagramas no componente Interface, calcular algumas estatísticas e descartar os datagramas (entende-se que os datagramas são repassados à camada superior ou simplesmente descartados pelo protocolo componente IP*), sendo que esta definição contempla o requisito F16.* Para realizar suas funcionalidades, depende das <<interfaces>> dos componentes Mensagem e Interface.

Uma situação especial da utilização deste componente é sua utilização nos componentes IP para viabilizar o descarte de datagramas.

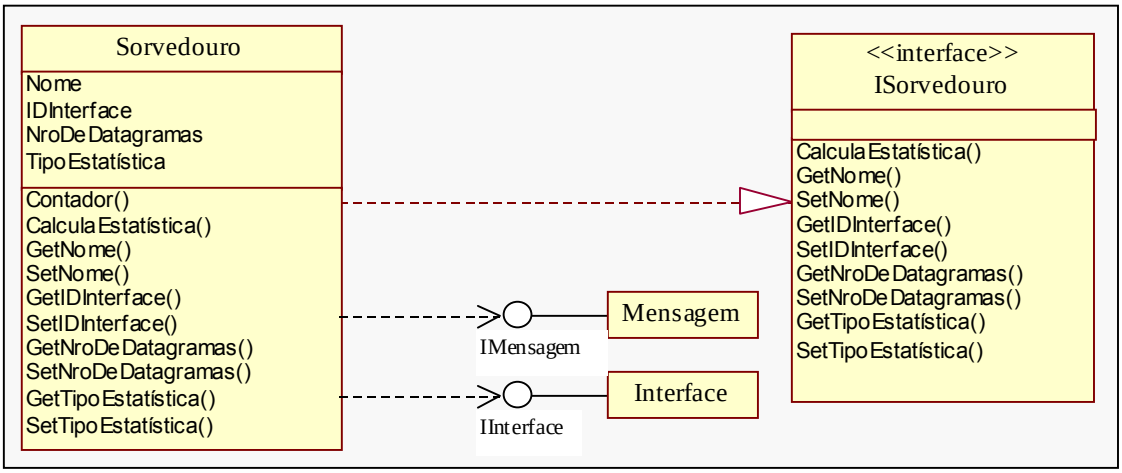


Figura 4.21: Componente Sorvedouro.

A figura 4.21 apresenta o componente Sorvedouro com suas devidas propriedades e operações, juntamente com a realização de sua <<interface>> (ISorvedouro)e suas dependências com relação a outros componentes. A seguir apresentam-se as descrições de suas propriedades e operações:

- ✓ *Nome: Identifica o componente Sorvedouro.* Permite operações Get e Set;
- ✓ *IDInterface*: identifica o componente *Interface* do componente *Host* pela qual ele está ligado. Permite operações Get e Set;
- ✓ *NroDeDatagramas*: número de datagramas descartados; Contém operações Get e Set;
- ✓ *TipoEstatística*: alguma estatística definida pelo usuário, como por exemplo, tempo médio de permanência de um datagrama no sistema. Permite operações Get e Set;
- ✓ *Contador*(): incrementa *NroDeDatagramas*;
- ✓ *CalculaEstatística*(): realiza os cálculos pertinentes a *TipoEstatística*.

4.4.2.11. Enlace

O componente Enlace representa o meio físico de transmissão existente entre componentes Interface contidos nos componentes de Hosts e Roteadores. Neste trabalho o Enlace é restrito a uma rede ponto-a-ponto *full-duplex*, sendo ligado a um componente Interface em cada extremidade do mesmo. Ele depende das <<interfaces>> dos componentes Mensagem e Interface.

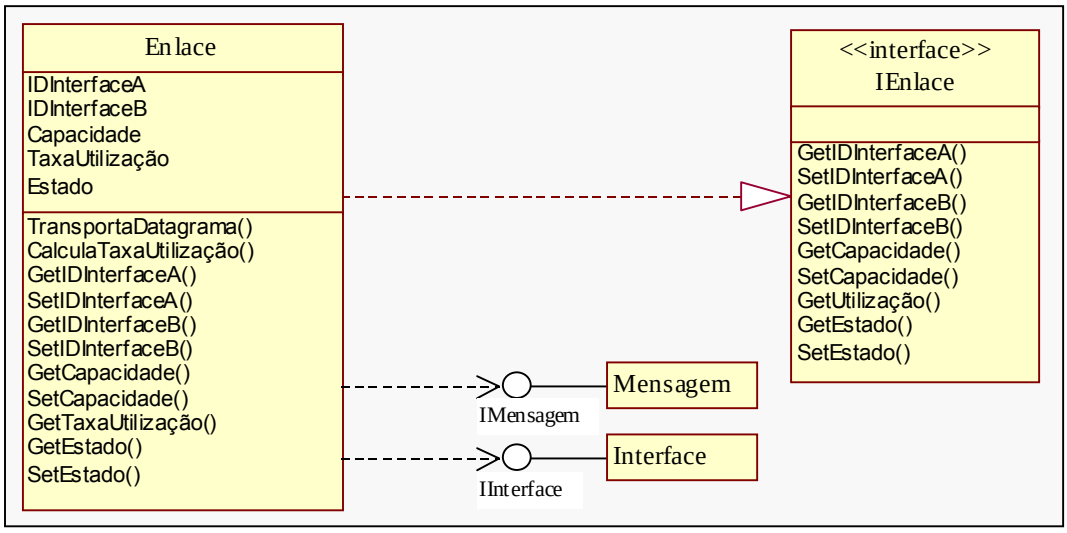


Figura 4.22: Componente Enlace.

A figura 4.22 apresenta o componente Enlace com suas devidas propriedades e operações, juntamente com a realização de sua interface (IEnlace) e suas dependências com relação aos serviços oferecidos por outros componentes. A seguir apresentam-se as descrições de suas propriedades e operações:

- ✓ *IDInterfaceA* a *IDInterfaceB*: identifica os componentes *Interfaces* pelos quais o componente *Enlace* está conectado. Permite operações Get e Set;
- ✓ *Capacidade*: número de bytes por segundos que podem ser transmitidos. Permite operações Get e Set;
- ✓ *TaxaUtilização*: medida de desempenho referente a percentagem de utilização do *Enlace*. Permite operações Get e Set;
- ✓ *Estado*: indica se o componente *Enlace* está ativo(congestionado ou disponível) ou inativo. Esta proposição contempla o requisito F15. Permite operações Get e Set;
- ✓ *TransportaDatagrama*(): função que busca um datagrama na *Interface* de origem e envia para *Interface* de destino;

4.4.2.12. Roteador

O Componente Roteador tem a função de repassar unidades de informação em uma rede TCP/IP, as quais trafegam desde um componente Host origem (geradas por um componente FonteTráfego) até um ou mais componentes Hosts destino (e descartadas por um componente Sorvedouro), passando por N componentes Roteador. Cada componente Roteador é composto por um componente CamadaInternet e 2+N componentes Interface. Desta forma o requisito **F3** é contemplado. Cada componente Interface liga o Roteador a outro componente Roteador ou a um componente Host, por intermédio de um componente Enlace. Este componente utiliza serviços oferecidos pelos componentes Interface e CamadaInternet. O componente Roteador deve ser configurado pelo usuário.

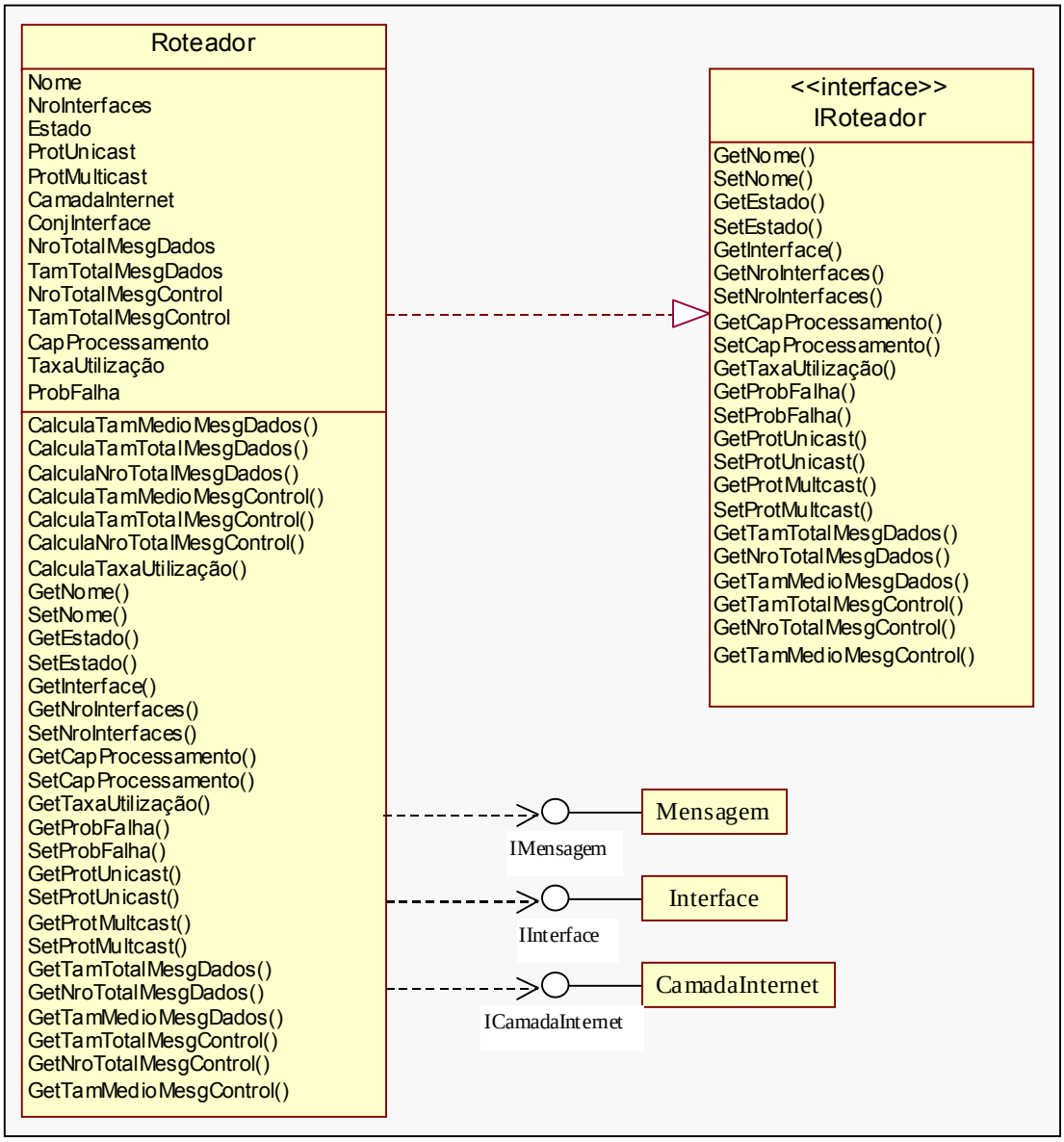


Figura 4.23: Componente Roteador.

A figura 4.23 apresenta o componente Roteador com suas devidas propriedades e operações, juntamente com a realização de sua <<interface>> (IRoteador) e suas dependências com relação aos serviços oferecidos por outros componentes. A seguir apresentam-se as descrições de suas propriedades e operações:

- ✓ Nome: identifica o componente Roteador. Permite operações *Get* e *Set;*
- ✓ NroInterfaces: Número de componentes Interfaces contidas no componente Roteador; Esta propriedade é informada pelo usuário. Permite operações *Get* e *Set;*
- ✓ Estado*: indica se o componente* Roteador *está ativo (ocupado ou disponível) ou inativo.* Esta proposição contempla o requisito F15 Permite operações *Get* e *Set;*
- ✓ ProtUnicast: define o protocolo de roteamento *unicast* a ser utilizado pelo componente Roteador, através de sua CamadaInernet. Esta propriedade é informada pelo usuário. Permite operações *Get* e *Set;* Esta definição contempla parcialmente o requisito F6;
- ✓ ProtMulticast: define o protocolo de roteamento *multicast* a ser utilizado pelo componente Roteador, através de sua CamadaInernet. Esta propriedade é informada pelo usuário. Permite operações *Get* e *Set;* Esta definição contempla parcialmente o requisito F6;
- ✓ CamadaInternet: *identifica um componente* CamadaInternet*, o qual pertencente ao componente* Host;
- ✓ ConjInterface: *identifica um conjunto, com* NroInterfaces *de componente* Interface*, o qual viabiliza a comunicação do componente* Roteador *com outros componentes* Roteador *e componentes* Host *por intermédio de um componente* Enlace*;*
- ✓ NroTotalMesgDados: Medida de desempenho que armazena o número total de datagramas de dados tratados pelo componente Roteador. Permite operações *Get* e Calcula;
- ✓ TamTotalMesgDados: Medida de desempenho que armazena a quantidade, em bytes, de datagramas de dados tratados pelo componente Roteador. Permite operações *Get* e *Set;*
- ✓ NroTotalMesgControl: Medida de desempenho que armazena o número total de datagramas de controle tratados pelo componente Roteador; Permite operações *Get* e Calcula;;
- ✓ TamTotalMesgControl: Medida de desempenho que armazena a quantidade, em de bytes de datagramas de controle tratados pelo componente Roteador. Permite operações *Get* e Calcula;
- ✓ CapProcessamento: define a capacidade de processamento do componente Roteador medida em megabits por segundo. Permite operações *Get* e *Set;*
- ✓ TaxaUtilização: Medida de desempenho que armazena a percentagem da capacidade do componente Roteador utilizada. Permite operações *Get* e *Set;*

- ✓ ProbFalha: propriedade de simulação que determina a probabilidade do Roteador ficar inativo durante a simulação. Permite operações *Get* e *Set;*
- ✓ CalculaTamMedioMesgDados(): cálculo da medida de desempenho pertinente ao tamanho médio das mensagens de dados tratadas pelo componente Roteador, obtida a partir da divisão de TamTotalMesgDados por NroTotalMesgDados;
- ✓ GetTamMedioMesgDados(): operação que disponibiliza a medida de desempenho pertinente ao tamanho médio das mensagens de Dados tratadas pelo componente Roteador;
- ✓ CalculaTamMedioMesgControle(): cálculo da medida de desempenho pertinente ao tamanho médio das mensagens de controle, obtida a partir da divisão de TamTotalMesgControle por NroTotalMesgControle;
- ✓ GetTamMedioMesgControle(): operação que disponibiliza a medida de desempenho pertinente ao tamanho médio das mensagens de controle tratadas pelo componente Roteador;

Pode-se observar que todos os componentes de alto nível, e alguns de baixo nível contém propriedades correspondentes a medidas de desempenho, o que busca contemplar o requisito **F14**.

4.5. Glossário de Conceitos Relevantes

Para garantir a transparência da comunicação entre as partes envolvidas no projeto é interessante que se elabore um glossário dos termos mais importantes. Este glossário é desenvolvido durante as várias fases do projeto, entretanto, ao realizar o diagrama de conceitos tem-se claro quais os termos em evidência [Larman, 98]. Segue o glossário deste trabalho:

Add-on	Protocolos que trabalham na camada internet para realizar funções extras as funcionalidades básicas. O RSVP (*Reservation Resource Protocol*) que realiza controle de congestionamento é um exemplo
BGMP	*Border Gateway Multicast Protocol* é um protocolo de roteamento dinâmico *multicast* que atua entre sistemas de roteamento.
BGP	*Border Gateway Protocol* é um protocolo de roteamento dinâmico que atua entre sistemas autônomos;
Broadcast	Um sistema de entrega de pacotes que entrega uma copia de um dado pacote para todos os *hosts* anexados a rede. A função *broadcast* pode ser implementada em hardware (ex. *ethernet*) ou em software (ex. *Cypress*) [Comer, 95]
Camada Internet	Camada pertencente a arquitetura TCP/IP responsável pela interconexão de redes;

CBT	O *Core Border Tree* é um protocolo de roteamento *multicast* que constrói apenas uma árvore de roteamento para cada grupo *multicast*.
Datagrama	Unidade de informação utilizada na camada internet. Um datagrama é composto por duas partes: cabeçalho e corpo do da informação. O corpo da informação contém a unidade de informação da camada superior. Também é chamado de Pacote.
Dupla Camada	É uma estratégia de transição do IPv4 para IPv6, a qual consiste da implementação de ambas as versões em um nodo.
DVMRP	O *Distance Vector Multicast Routing Protocol* é um protocolo de roteamento *multicast* que atua sobre o protocolo de roteamento *unicast* RIP e atua em um sistema autônomo.
EGP	*Exterior Gateway Protocol*, corresponde ao tipo de protocolos que atuam entre sistemas autônomos;
Endereçamento	Esquema que define o formato e significado de endereços que identificam as interfaces de cada equipamento e grupos *multicast* pertencentes a uma rede TCP/IP. O IPv4, a princípio, tem o esquema de endereçamento baseado em 5 classe e recentemente passou a utiliza um esquema sem classes (CIDR).
Enlace	Meio físico de transmissão de informações entre 2 ou mais equipamentos de uma rede. Um enlace pode ser ponto a ponto (um equipamento em cada extremidade) ou multiponto (vários equipamentos conectados ao meio). Um equipamento é ligado ao enlace por intermédio de uma interface.
Fila	Estrutura de armazenamento temporário de Informações (Datagramas). Um equipamento de rede(roteador, *host*) contém uma única Fila de Entrada (F_Entrada) e uma Fila de saída (F_Saída) para cada interface.
Forward	É o algoritmo que consulta a tabela de roteamento e decide por qual(is) interface(s) deve enviar o datagrama. São considerados 3 algoritmos, de acordo com o tipo de comunicação desejada: *Forward Unicast, Forward Multicast*, e *Forward Broadcast*, sedo que este último existe apenas no IPv4;
Fragmentação	Função que divide um datagrama em vários datagramas. Desfragmentação é a função inversa;
***Host*:**	algum sistema de computador do usuário final que está conectado a uma rede. A faixa de tamanho de um *host* varia desde um computador pessoal (PC) até supercomputadores[Comer, 95].

ICMPv4	*Internet Control Message Protocol* especifica as mensagens de controle e de erros utilizadas no IPv4 para comunicação de duas instâncias da camada IP. O v4 corresponde a versão do IPv4, onde atua, e não a versão do ICMP;
ICMPv6	*Internet Control Message Protocol* especifica as mensagens de controle e de erros, inclusive controle de gerenciamento de grupos *multicast*, no IPv6;
IGMP	*Internet Group Management Protocol* é o protocolo responsável por gerenciar grupos *multicast* no IPv4;
IGP	*Internal Gateway Protocol*, corresponde ao tipo de protocolos de roteamento que atuam no interior de um sistema autônomo;
Interface	Ligação entre hardwares ou softwares
Internet Protocol	O IP, é o protocolo responsável por proporcionar a interconexão de redes. O IPv4 é sua versão mais utilizada atualmente, entretanto está sendo gradualmente substituída pelo IPv6.
Internetwork	Conjunto de redes interligadas, formando uma grande rede. Também chamada de internet. O maior exemplo é a Internet.
Loopback	É o endereço usado por um sistema que deseja enviar uma mensagem para ele mesmo, onde o protocolo IP recebe a mensagem do protocolo da camada de transporte e devolve a ele como se ela estivesse vindo de outro computador.
Mensagens de Controle e de Erros	Mensagens utilizadas para que duas instâncias da camada *internet* se comuniquem. Estas mensagens são empacotadas em um datagrama, para transmissão.
MOSPF	O *Multicast Extension Open Shortest Path Firts* é um protocolo de roteamento *multicast* que atua sobre o protocolo de roteamento *unicast* OSPF e pode atuar tanto em ou entre sistemas autônomos.
OSPF	O *Open Shortest Path First* é um protocolo de roteamento *multicast* baseado no algoritmo Estado de Enlace;
PIM-DM	O *Protocol Independent Multicast – Dense Mode* é um protocolo de roteamento *multicast* designado a atuar em um sistema autônomo, independentemente do protocolo de roteamento *unicast*;
PIM-SM	O *Protocol Independent Multicast – Sparse Mode* é um protocolo de roteamento *multicast* designado a atuar entre sistemas autônomos, independentemente do protocolo de roteamento *unicast*
Protocolo de	É o protocolo responsável por criar e manter as tabelas de roteamento;

Roteamento	
Protocolo *Multicast*	Protocolo responsável por definir as rotas para um grupo *multicast*
RIP	*Routing Internet Protocol*, antigo protocolo de roteamento *unicast* baseado no algoritmo Vetor de Distância;
Roteador	um computador dedicado, de propósito especial que liga duas ou mais redes e repassa pacotes de uma para outra. Em particular, um roteador IP repassa datagramas IP através de redes, as quais ele esta conectado. Um roteador utiliza o endereço destino em um datagrama para escolher o próximo salto para o qual ele repassará o datagrama. Originalmente os pesquisadores o chamavam de *gateway* IP [Comer, 95].
Roteamento *Unicast*	Categoria de protocolos de roteamento que enviam mensagens *unicast*;
Sistema Autônomo	Conjunto de redes sob um único domínio administrativo;
Tabela de Roteamento	É a base de dados que armazena as melhores rotas para cada destino(*host* ou rede) possível em uma rede. Existe Tabela *Unicast* e a Tabela *Multicast*, cada uma gerada pelos respectivos protocolos. A tabela *Multicast* muitas vezes é gerada sob demanda e mantida apenas na memória volátil;
Transição	Mecanismo (estratégia) de transição definido para permitir a interoperabilidade entre equipamentos IPv6 e IPv4;

Capítulo 5

5. Conclusões e Sugestões

Este capítulo apresenta as conclusões do presente trabalho como também sugestões de sua continuidade.

5.1. Conclusões

Neste trabalho foi apresentado uma especificação de componentes de software genéricos, que representam as funcionalidades mínimas dos elementos necessários para modelagem de uma rede TCP/IP. Essa especificação foi realizada utilizando a Linguagem de Modelagem Unificada (UML) seguindo um processo de desenvolvimento apresentado em [Larman,98], enfocando a fase planejar e elaborar, a fase de análise e a fase de projeto de alto nível (projeto arquitetural). Em adicional, juntamente com o projeto arquitetural, apresentou-se os componentes, detalhadamente, em forma de classe com suas devidas <<interfaces>>. O nível de detalhamento de alguns componentes chegou a um nível em que suas propriedades e operações, respectivamente, poderão ser diretamente mapeados para atributos e métodos de classe.

No decorrer deste trabalho desenvolveu-se um estudo sobre técnicas de avaliação de desempenho, focando os esforços em ambientes de simulação digital. Parte deste estudo é apresentado na seção 2.5, e um estudo mais detalhado pode ser encontrado em [Wagner, 00] publicado na Semana de Informática da Universidade Federal da Bahia, o qual avalia a viabilidade de se utilizar o ambiente Arena na avaliação de redes de computadores. Com isto, acredita-se ter

atingido o objetivo específico de estudar a técnica da simulação digital voltadas para a avaliação de desempenho de redes de computadores.

Segundo [D'Souza], ao realizar uma especificação de um modelo deve-se definir claramente qual o nível de abstração que se deseja atingir. No estudo realizado sobre tecnologia TCP/IP, centrou-se principalmente no contexto das *internetworks* (camada internet), abordando os mecanismos que as viabilizam. Nesta área de estudo existem vários trabalhos em andamento na IETF (órgão responsável pela padronização dos protocolos da Internet), tais como desenvolvimento de protocolos de roteamento *unicast* e *multicast*, a questão da transição do IPv4 para o IPv6, protocolos de controle de congestionamento, entre outros. Como resultado deste estudo, chegou-se a um minucioso tutorial, apresentado no capítulo 3, o qual detalha as funcionalidades e protocolos que envolvem a camada internet. Um estudo prévio, sobre arquiteturas de redes foi realizado, entretanto entendeu-se ser conveniente disponibilizar (no apêndice A), uma breve introdução a este assunto, e descrever sucintamente a composição da arquitetura TCP/IP. E como estudo complementar ao capítulo 3, foi disponibilizado (nos apêndices B e C) um material referente aos principais protocolos e algoritmos de roteamento dinâmico definidos pela IETF. Desta forma, acredita-se ter atingido os objetivos específicos de estudar as arquiteturas de redes de computadores e detalhar as funcionalidades da camada internet da arquitetura TCP/IP.

Mesmo centrando o escopo do estudo à camada internet da arquitetura TCP/IP, percebeu-se ser praticamente impossível elaborar um modelo de simulação que englobe todos seus aspectos. No entanto, constatou-se a existência de funcionalidades básicas que devem constar em qualquer modelo desta natureza. Partindo desta premissa, investiu-se em uma especificação dessas funcionalidades, a qual servirá de ponto de partida para modelos mais específicos. Desta forma, o analista de modelagem com essa especificação, poderá refiná-la rapidamente, e centrar suas atenções na modelagem de seu objeto de estudo. Para realizar esta especificação foi necessário a utilização de técnicas de engenharia de software para permitir, com eficácia, a reusabilidade da especificação até então pretendida. Conceitos prévios, de um estudo sobre da orientação a objetos foram apresentados na seção 2.4, e estudos mais detalhados sobre o processo de desenvolvimento e notação da especificação foram apresentados, sob demanda, no capítulo 4. Desta forma, acredita-se, ter atingido o objetivo específico pertinente ao estudo de técnicas utilizadas pela engenharia de software para desenvolvimento de componentes de software.

O objetivo geral desta Dissertação de mestrado: "uma especificação de componentes voltados para a modelagem de redes TCP/IP"; foi alcançado com a elaboração do capítulo 4, o qual descreve em detalhes o: Domínio do Problema, a Fase de Análise, a Fase de projeto de alto nível e Glossário de Conceitos Relevantes.

Nessa especificação teve-se o cuidado de certificar-se dos quesitos necessários para a elaboração de um bom documento, o qual, segundo [D'Souza, 98], pode-se intercalar narrativas, figuras, diagramas, tabelas, etc., além de um glossário dos termos mais representativos.

Quanto à primeira parte do documento, destaca-se a elaboração do diagrama de *use-case* da UML, o qual não se limita a mostrar somente as funcionalidades dos componentes propostos. Nesse diagrama também é mostrado, em alto nível de abstração, a funcionalidade de um ambiente de simulação no qual os componentes podem estar inseridos. Este diagrama, possibilita uma visão geral do escopo do trabalho necessária em todas as fases de desenvolvimento dessa especificação.

Quanto à elaboração da documentação pertinente à Fase de Análise, chama-se a atenção para a complexidade e a grande abrangência da tecnologia TCP/IP. A constatação desse fato leva, de início, a uma simplificação na relação de requisitos considerados evidentes. Essa constatação fica particularmente ressaltada na construção dos diversos diagramas apresentados no capítulo 4. Inicialmente, o diagrama de conceitos, que numa primeira versão mais abrangente aponta para uma grande quantidade de conceitos e, consequentemente, para uma grande quantidade de operações relacionadas com os componentes inicialmente propostos. Uma simplificação desse modelo evidenciou os conceitos mais relevantes dessa tecnologia. Ainda, dois diagramas de seqüência relacionados aos *use-cases* "Receber e Repassar Datagramas" e "Criar e Manter Tabelas de Roteamento" ressaltam também essa constatação. Um diagrama de estados também foi elaborado para destacar os principais eventos, estados e transições pertinentes ao ciclo de vida do Roteador.

Ao realizar a documentação relacionada a Fase de Projeto (Arquitetural), constatou-se que a notação UML ainda não se apresenta adequada o suficiente para representar sem ambigüidade um componente de alto nível. No entanto, utilizou-se os artefatos "pacote", "classe" e "interface" para especificar cada componente, de forma que se possa identificar explicitamente os serviços oferecidos e requeridos por um componente.

Na primeira parte da especificação definiu-se dois tipos de usuários: o Analista de Modelagem e o Desenvolvedor de Softwares de Simulação. Portanto, teve-se o cuidado de tratar aspectos relacionados a reusabilidade deste documento. Acredita-se ter alcançado um bom grau de reusabilidade, principalmente pela utilização de técnicas orientadas a componentes, como a definição de um *framework* de componentes para a camada internet, isolando os componentes passíveis de alteração e/ou acoplamento de novas funcionalidades. Outro aspecto que ressaltou a reusabilidade foi a utilização da UML, linguagem esta, de modelagem, que está se difundindo mundialmente, no campo da informática.

Essa especificação de componente também poderá ser reutilizada em projetos de ambientes de simulação orientados a componentes voltados para a modelagem e avaliação de desempenho de redes de computadores, como também, poderá ser reutilizada em ambientes de simulação de alto nível como o Arena, permitindo que novos componentes sejam a ele integrados.

Esta afirmativa se justifica pelo cuidado que se teve em elaborar uma documentação rica em figuras, diagramas, tabelas e narrativas explicativas, como sugere [D'Souza, 98].

5.2. Sugestões para trabalhos futuros

A especificação de componentes ora apresentada é ponto de partida para trabalhos relacionados com reutilização da especificação de software, utilizando-se de artefatos de software, tais como *frameworks* de componentes voltados para o desenvolvimento de ferramentas de simulação de redes com a tecnologia TCP/IP.

Uma sugestão convidativa de continuidade dessa Dissertação é investir no refinamento da especificação do *framework* de componentes definido neste trabalho, na especificação do componente Camada Internet. Outros *frameworks* de componentes poderiam ser especificados, para atender problemas mais específicos, e serem agregados a este *framework (*Camada Internet*)*, como por exemplo a especificação um *framework* de componentes que abstraia as funcionalidade dos protocolos de roteamento *multicast*. Esses *frameworks* permitiriam obter um conjunto de componentes distintos a partir de um esforço menor do que o necessário para desenvolver o modelo de cada um isoladamente.

Uma evolução natural deste trabalho, corresponde ao refinamento desta especificação concentrada em elaborar uma estratégia melhor para coleta de medidas de desempenho a fim de isolar as medidas de desempenho da lógica do problema. Uma alternativa seria agregar, aos componentes específicos do domínio do problema (Roteador, Interface, IP, etc.), um componente chamado *MedidaDesempenho*.

Outra sugestão volta-se a criação de um painel sendo parte de um *template* de componentes que representem elementos de uma rede TCP/IP que possa ser integrado ao ambiente Arena.

Um estudo mais detalhado da tecnologia TCP/IP pode levar naturalmente a proposta de novos componentes que representem com maior fidelidade recursos e mecanismos dessa tecnologia.

Finalmente, a especificação ora apresentada, pode ser ponto de partida para a construção de um ambiente de simulação específico para a modelagem e avaliação de desempenho de redes de computadores. O diagrama de *use-case* apresentado no capítulo 4, ilustra a funcionalidade desse ambiente.

6. Referências Bibliográficas

[Alaettinoglu, 98] **Alaettinoglu, C.; shankar, A.; Dussa-Zieger, K. Matta, I.;** *"Design and Implementation of MaRS: A Routing Testbed"*; Journal of Internetworking: Research & Experience"; vol. 5 nº 1, 17-41, 1994.

[Almeida, 99] **Almeida, Marcelo, J. S. C.; "***ATMLib – Uma Biblioteca de Classes para a Construção de Simuladores de Redes ATM: Proposta e Implementação*", Dissertação de Mestrado, CCT, Universidade Federal da Paraíba, Campina Grande, novembro de 1999.

[Ballardie, 97] **Ballardie, A.;** *"Core Based Trees (CBT version 2) Multicast Routing"*; RFC 2189, september, 1997.

[Baran, 64] **Baran P.;** *"On Distributed Communication Networks",* IEEE Trans. On Commun. Systems, vol. CS-12, pp. 1-9, março de 1964.

[Barcellos, 99] **Barcellos, Marinho;** *"Comunicação Multicast na Internet",* Minicurso 4, 17º Simpósio Brasileiro de Redes de Computadores, Salvador-BA, maio de 1999.

[Cabral, 90] **Cabral, M. I. C.;"***SAVAD - Curso de Avaliação de Desempenho em Redes de Computadores",* I JORNNAI - UFPB/IBM, Campina Grande, 1990.

[Cabral, 92] **Cabral, M. I. C.;"***Um Ambiente de Simulação Inteligente para avaliar o desempenho de Sistemas Distribuídos",* Projeto de Pesquisa, CNPq/UFPB, Campina Grande, 1992.

[Cadence, 98] **Cadence Inc.;** *"BONeS Simulator";* http://www.cadence.com/alta/products/bonesdat.html, 1998.

[Castineyra, 96] **Castineyra, I., Chiappa, J. N., Steenstrup, M.;** *"The Nimrod Routing Architecture",* Internet-Draft, fevereiro de 1996.

[Celestino, 90] **Celestino, J.; "***Avaliação de Desempenho em Redes Locais Brasileiras",* Dissertação de Mestrado, CCT, Universidade Federal da Paraíba, Campina Grande, fevereiro de 1992.

[Chiappa, 94] **Chiappa, J. N.;** *"Ipng Technical Requirements Of the Nimrod Routing and Addressing Architecture",* RFC 1753, dezembro de 1994.

[Chu, 78] **Chu, K.;** *"A Distribuided Protocol for Updatin Network Topology Information",* Report RC 7235, IBM T. J. Watson Research Centrer, 1978.

[Cisco, 98] **Cisco System Inc.;** *"OSI Routing",*. Cisco Documentation - Technology Information. On-Line, Capturado em 10 de maio de 1998, http://www.cisco.com/univercd/home/home.htm.

[Cisco, 98b] **Cisco System Inc.;** *"OSPF-2 Protocol overview",* Cisco Documentation – Technology Information. On-Line, Capturado em 10 de maio de 1998, http://www.cisco.com/univercd/home/home.htm.

[Cisco, 98c] **Cisco System Inc.;** *"BGP-4 Protocol overview* Cisco Documentation-Technology Information. On-Line, Capturado em 10 de maio de

1998,http://www.cisco.com/univercd/home/home.htm.

[Clocksin, 81] **Clocksin, W. F.; Mellish, C. S**.; *"Programming in Prolog"*. Berlim: Springer-Verlag, 1981.

[Coltun, 99] **Coltun, R., Moy, John T.;** *"OSPF Version 2 For IP Version 6",*. RFC 2740, dezembro de 1999.

[Comer, 94] **Comer, Douglas;** *"Internetworking with TCP/IP, Vol. 2: Design, Implementation, and Internals",*. Prentice Hall, 3ª Edição, 1995.

[Comer, 95] **Comer, Douglas;** *"Internetworking with TCP/IP, Vol. 1: Principles, Protocols, and Architecture",*. Prentice Hall, 3ª Edição, 1995.

[Conceição, 93] **Conceição F. H.C, M.I.C.;"***SIM/SAVAD - Um Simulador de Modelos de Redes de Filas"* Dissertação de Mestrado, CCT, Universidade Federal da Paraíba, Campina Grande, dezembro de 1993.

[Conta, 98] **Conta, A.; Deering, S.;** *"Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification"*; RFC 2463, December, 1998.

[D'Souza, 98] **D'Souza Desmond, F.; Wills, Alan C.;** *"Objects, Components and Frameworks with UML: The Catalysis Approach"*; Addison-Wesley, 1998.

[Damoun, 79] **Damoun, F., Kleinrock, L.;** *"Stochastic Performance Evaluation of Hierarchical Routing for Large Networks",* Computer Networks, vol. 3. Pp. 337-353, novembro de 1979.

[Deering, 98] **Deering, S.;***"Host Extension for IP Multicast.",* RFC 1112, May 1998.

[Deering, 98b] **Deering S. Hinden, R. J.;***"Internet Protocol, Version 6 (IPv6) - Specification",* RFC 2460, December 1998.

[Deering, 98c] **Deering, S. Hinden, R. J.;***"IP Version 6 Addressing Architecture",* RFC 2373, July 1998.

[Dias, 92] **Dias, Maria Madalena;"***SIMILE - Um Simulador Reutilizável para Avaliação de Desempenho de Redes Locais"* Dissertação de Mestrado, CCT, Universidade Federal da Paraíba, Campina Grande, abril de 1992.

[Dijkstra, 59] **Dijkstra, E. W.;** *"As A Note on Two Problems in Connection with Graphs",* Numer. Math., vol. 1. Pp. 269-271, outubro de 1959.

[Englander, 97] **Englander, Robert***. "Developing Java Beans"***,** O'Reilly & Associates, Inc., 1997.

[Fenner, 97a] **Fenner W.;** *"Internet Group Management Protocol, Version 2",* RFC 2236, novembro, 1997.

[Fowler, 97] **Fowler, Martin; Scott, Kendall.** *UML Distilled***,** Addison-Wesley, 1997.

[Fonseca, 99] **Fonseca, Ana C.;** *"Código Morse – A Hora da Aposentadoria", Revista Época, Editora Globo, Edição. 37, 01 de fevereiro de 1999.*

[Ford, 62] **Ford, L. R. Jr., Fulkerson, D. R.;** *"Flows in Networks",* Princeton University Press, Princeton, N. J., 1962.

[Freire, 00] **Freire, Raissa Dantas;** *"Especificação de um Framework baseado em Componentes de Software Reutilizáveis para Aplicações de Gerência de Falhas em Redes de Computadores";* Dissertação de Mestrado, CCT, Universidade Federal da Paraíba, Campina Grande, abril de 2000.

[Fuller, 93] **Fuller, V.; Li, T.; Varadhan, K.;** *"Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy"***;** RFC 1519, September, 1993

[Gilligan, 00] **Gillgan, R., Nordamark, E.;** *"Transition Mechanisms for Ipv6 Hosts and Routers"* . <draft-ietf-ngtrans-mech-06.txt> of Obsoletes RFC 1933, 14 april, 2000.

[Helmy, 98] **Helmy, A.; Deering, S.; Farinacci, D.; Jacobson, V.; Estrin, D.; Wei, L.;** *"Protocol Independent Multicast version 2, Dense Mode Specification"*; <draft-ietf-pim-v2-dm-01.txt>, novembro 1998.

[Helmy, 99] **Helmy, A.; Deering, S.; Farinacci, D.; Jacobson, V.; Estrin, D.; Wei, L.;** *"Protocol Independent Multicast Sparse Mode (PIM-SM): Protocol Specification"*; <draft-ietf-idmr-pim-sm-specv2-00.txt> Expirou em maio de 2000, Extraído on-line junho de 2000; novembro de 1999.

[Helton, 98] **Helton, D.;** *"The Impact of large-scale component and framework application development on business";* In Third International Workshop on Component-Oriented Programming (WCOP'98), Brussels, 1998.

[Hendrick, 88] **Hendrick, Charles;** *"Routing Information Protocol",* Rutgers University, RFC-1058, junho de 1988.

[Huitema, 96] **Huitema, Christian;** *"IPv6, The New Iinternet Protocol",* Prentice Hall, ISBN 0-13-241936-X, 1996.

[ISO, 92] **International Organization for Standarization / International Eletrotechnical Committee;** *"Information Technology - Open Systems Interconnection Reference Model - Part 1: Basic Reference Model (Revision of First Edition - ISO 7498:1994)",* Draft International Standard ISO/IEC DIS 7498-1, 1992.

[Johnson, 97] **Johnson Vicki, Jonhson Marjory***; "Introduction IP Multicast Routing"***;** IP Multicast Iciciative; Startdust Forums Inc. Extraído da Internet www.ipmulticast.com em janeiro de 1999. Camppbell, CA USA, 1995-1997.

[Kelton, 98] **Kelton, W. David; "***Simulation with arena"*, WCB/McGraw-Hill; 1998.

[Kessler, 92] **Kessler, Andrew S.;** *"An Analysis of Dynamic Routing Protocols in Highly Distributed Data Networks",* Thesis requirement for the degree Master of Science Interdisciplinary Telecommunications Department of the Faculty of the Graduate School of the University of Cologrado, dezembro de 1992.

[Kronbauer, 98] **Kronbauer, Artur H.; "***Avaliação de Protocolos Multicast em Redes TCP/IP",* Dissertação de Mestrado, CCT, Universidade Federal da Paraíba, Campina Grande, junho de 1998.

[Landin, 98] **Landin, N., Niklasson, A.;** *"Development of Object-Oriented Frameworks"***;** Department of Communication Systems, Lund Institute of Technology, Sweden, 1998.

[Larman, 98] **Larman, C;** *"Applying UML and Patterns – Na Introduction to Object-Oriented Analysis and Design"*; Prentice-Hall, 1998.

[Lobão, 96] **Lobão Edílio de C., Porto José V.; "***Proposta para Sistematização de Estudos de Simulação",* Artigo, Escola de Engenharia de São Carlos, 1996.

[Lula, 00] **Lula, Juliana C. L de A,** *"Desenvolvimento de Componentes para Simuladores de Redes de Computadores"***,** Projeto de Dissertação de Mestrado, Pós-Graduação em Informática, UFPB, abril de 2000.

[Malkin, 97] **Malkin, Gary, Minner, R.;** *"RIPng for IPv6",* RFC 2080, novembro de 1997.

[Moy, 94] **Moy, J.;** *"Multicast Extension to OSPF"*; Disciplina do curso RFC 1584, march 1994.

[Moy, 97] **Moy, John T.;** *"OSPF Version 2",* Cascade Communications Corp., draft-ietf-ospf-version2-11.txt, outubro de 1997.

[Moy, 98] **Moy, John T.;** *"OSPF : Anatomy of an Internet Routing Protocol",* Addison Wesley Longman, Inc; maio de 1998.

[ns, 98] **ns;** *"Network Simulator";* http://www-mash.cs.berkeley.edu/ns, 1998.

[Oliveira, 95] **Oliveira, S. R DE M.;** *"ALLOS - Uma Ferramenta para Solucionar Modelos de Redes de Filas Usando Cadeias de Markov"*; Dissertação de Mestrado. CCT, Universidade Federal da Paraíba, junho de 1995.

[OMG, 00] **OMG, Object Management Group** *"OMG Unified Modeling Linguage Specification***";** on-line http://cgi.omg.org/cgi-bin/doc?formal/00-03-01 version 1.3, First Edition march 2000.

[Pegdrm, 95] **Pegdrm, C.D, Shannom, R. E.,Sadowski, R;***"Introduction to Simulation Using SIMAN"*; Mc-Graw-Hill, Inc., 1995.

[Postel, 81] **Postel, J.;***"Internet Protocol - Specification",* RFC 791, September 1981.

[Postel, 81b] **Postel, J., Reynolds, J.;***"Assigned Numbers",* RFC 1700, See also http://www.iana.org/numbers.html;October 1994.

[Pusateri, 99] **Pusateri, T.;***"Distance Vector Multicast Routing Protocol – Version 3"***,** IETF Draft draft-ietf-idmr-dvmrp-v3-09.txt, september, 1999.

[Ramanathan, 96] **Ramanathan, R., Steeenstrup, M.;** *"Nimrod Functionality and Protocol Specifications, Version 1",* Internet-Draft, março de 1996.

[Rekhter, 93] **Rekhter, Y.; Li, T.;** *"Na Architecture for IP Address Allcation with CIDR";* RFC 1518, september 1993.

[Roberts, 94] **Roberts, Chell; Dessouky, Yasser;** *"Na Overview of Object-Oriented Simulation"*; SIMULATION, Number 70, pp. 359-368, june, 1998.

[Rudin, 76] **Rudin, H.;** *"On Routing and Delta Routing: A Taxonomy and Performance Comparison of Techniques for Packet-Switched Networks",* IEEE Trans. On Commun., vol. COM-24, pp. 43-59, janeiro de 1976.

[Rumbaugh, 94] **Rumbaugh, James; Blaha Michael; Premerlani, William; Eddy, Fredirck; Lorensen Willian;** *"Modelagem e Projetos Baseados em Objetos"*; Editora Campus, 1994.

[Rumbaugh, 99] **Rumbaugh, James; Booch, Grady; Jacobson, Ivar** *"The Unified Modeling Language User Guide*"; Editora Addison-Wesley, 1999.

[Rumbaugh, 99b] **Rumbaugh, James; Booch, Grady; Jacobson,** *Ivar "The Unified Modeling Language Reference Manual"*; Editora Addison-Wesley, 1999.

[Sauvé, 00] **Sauvé, Jacques;** *"Análise e Projeto de Sistemas Orientados a Objeto"*; Disciplina do curso de Pós Graduação em Informática; Universidade Federal da Paraíba; Material on-line http://vulcano.dsc.ufpb.br/jacques/cursos/2000.1/apoo/; período 1º semestre, ano 2000.

[Sauvé, 86] **Sauvé, Jacques, P.; Beltrão, José Antão M. ; Giozza, Willian F.; Araújo, José Fábio M.;** *"Redes Locais de Computadores: Protocolos de Alto Nível e Avaliação de Desempenho"*; McGraw-Hill/Embratel, são Paulo-SP, 1986.

[Schimitt, 96] **Schimitt, A.**; "*Roteamento*". Curso de Roteamento. Universidade Federal

do Rio Grande do Sul, Porto Alegre, agosto de 1996.

[Schweitzer, 96b] **Schweitzer, Christiane M.**; "*Desenvolvimento de Baselines com o Uso de Simulação para Automação da Gerência de Redes*". Monografia de Conclusão do Curso de Ciência da Computação, UFSC, Florianópolis - SC, dezembro de 1996.

[Silva, 00] **Silva, Ricardo Pereira e;** . *"Suporte ao Desenvolvimento e Uso de Frameworks e Componentes*"; Tese de Doutorado, UFRGS/II/PPGC, Porto Alegre: março de 2000.

[Silva, 99] **Silva, Ricardo Pereira e., Price, Roberto Tom** *"Suporte ao desenvolvimento e uso de componentes flexíveis"*; Anias do XIII Simpósio Brasileiro de Engenharia de Software. Florianópolis, SC. outubro de 1999,

[Soares, 95] **Soares, Luiz. F. G., Lemos, G., Colcher, S.;** *"Redes de Computadores das LANs, MANs e WAN às Redes ATM"*, Ed. Campus, Rio de Janeiro, 1995.

[Soares, 97b] **Soares, Alex;** ***"****Roteamento: O que é Importante Saber"*, RNP News Generation vol.1, Nº1; http://www.rnp.br/newsgen/ascii/n1.txt. Maio de 1997.

[Souto, 93] **Souto, F.A.C.;** ***"****SAVAD – Sistema de Avaliação de Desempenho de Modelos de Redes de Filas*", Dissertação de Mestrado, CCT, Universidade Federal da Paraíba, Campina Grande, novembro de 1993.

[Stallings, 98] **Stallings, William;** *"High-Speed Networks: TCP/IP and ATM Design Principles*"; Prentice-Hall, Inc. New Jersey, 1998.

[Takus 97] **Takus, David A***. "Arena Software Tutorial"***,** Pensylvania, System Modeling Corporation, 1997.

[Tanenbaum, 97] **Tanenbaum, A.;** "*Redes de Computadores*"; Editora Campus; 3ª edição, 1997.

[Tarouco, 96] **Tarouco Liane**; "*CIDR-Classless Inter-Domain Routing*". Repositório de Documentos Técnicos do Grupo de Redes da Universidade Federal do Rio Grande do Sul, Porto Alegre, agosto de 1996. Extraído da Internet www.penta.ufrgs.br/cidr/tutorial.html.

[Thaler, 00] **Thaler, D.; Estrin, D.; Meyer, D.;** *"Border Gateway Multicast Protocol (BGMP): Protocol Specification"*; _____<draft-ietf-bgmp-spec-01.txt>, Expira em setembro de 2000; março de 2000.

[Thomas, 96] **Thomas, Stephen A.;** "*IPng and the TCP/IP Protocols – Implementing the Next Generation Internet*"; Ed. John Wiley & Sons, Inc. 1996.

[Wagner, 00] **Wagner, Marcus Vinícius; Alves, Carlos Alessander;** *Avaliação do Ambiente Arena para a Construção de Simuladores de Redes ATM*; VIII SEMINFO – Semana de Informática da UFBA; Salvador-BA, maio de 2000.

[Weck, 96] **Weck, W.** *"Independently extensible component Frameworks"* In: First International Workshop on Component-Oriented Programing (WCOP'96), Linz, 1996.

[Wessman, 96] **Wessman Petri;** *"Existing Routing Protocols and IPv6*"; Akumiitti Oy. Http://www.akumiitti.fi/~wessman. Finlândia, 1996.

[Zortech, 90] **Zortech Incorporated**.;***"****C++ Compiler Version 2.1".* Massachusetts, 1990.

Apêndice A

A. Arquitetura de Redes TCP/IP

Devido à gama de aspectos envolvidos na composição de uma rede de computadores sentiu-se a necessidade do desenvolvimento de mecanismos que reduzissem tamanha complexidade. Uma “Arquitetura de Redes de Computadores é composta por princípios, normas e técnicas estruturadas em N camadas hierárquicas definidas a fim de diminuir a complexidade dos projetos de redes”. Cada camada representa um programa ou processo, implementado em hardware ou software, utilizando as funções e serviços oferecidos pelas camadas inferiores [Soares, 95]. Como mostra a figura A.1, a camada N de uma máquina se comunica com a camada N em outra máquina, através de serviços oferecidos pela camada n-1. As regras e formatos que governam a conversação em uma camada ***i*** qualquer são chamados de protocolo da camada ***i***.

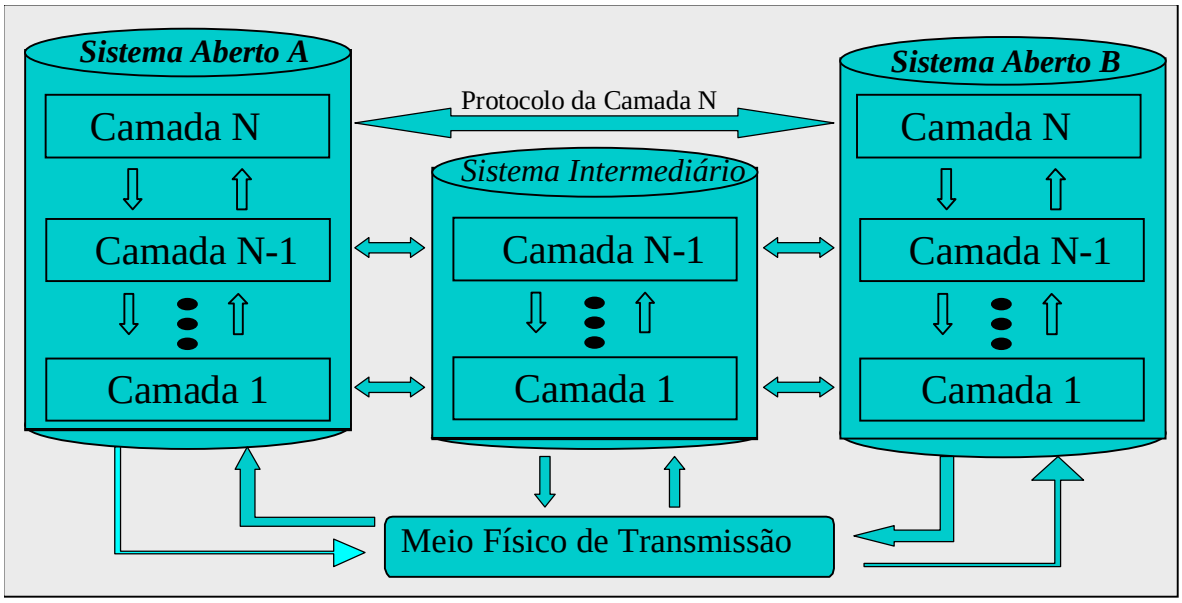


Figura A.1: Arquitetura de Redes de Computadores.

As arquiteturas podem ser classificadas em *arquiteturas proprietárias*, cujos detalhes são de conhecimento apenas de seu proprietário (fabricante) e *arquiteturas abertas*, controladas por órgãos padronizadores e disponíveis ao público em geral. As especificações de uma arquitetura aberta são expressas em documentos (padrões), os quais devem ser seguidos pelos fabricantes dos mais diversos componentes de redes (hardware e software) para possibilitar a interação de seus produtos com os produtos de outros fabricantes.

As arquiteturas abertas mais conhecidas são RM-OSI[9] e a Arquitetura Internet (TCP/IP). Seguem resumos da Arquitetura TCP/IP e da padronização Internet.

A.1. Arqui*t*etura TCP/IP

A arquitetura TCP/IP, também conhecida como arquitetura Internet, ou ainda suite Internet TCP/IP, define a interconexão entre redes de computadores, independente de suas tecnologias. O mecanismo que interconecta tais redes são chamados de *gateway IP* (roteador IP) [Comer, 95]. Um roteador IP (neste documento tal equipamento será reportado apenas pelo nome de roteador) teoricamente não implementa as camadas de alto nível, entretanto é possível utilizar um computador com duas (ou mais) interfaces de rede (placas de rede). No entanto, do ponto de vista do usuário, cada computador do usuário (*host*) está ligado a uma única grande rede, ficando toda complexidade, que envolve as implementações das redes, escondida.

A arquitetura TCP/IP é considerada um padrão de *facto*, pois é a principal arquitetura adotada no mundo todo para interconexão de redes. A arquitetura TCP/IP é composta por 4 camadas[Comer, 95]:

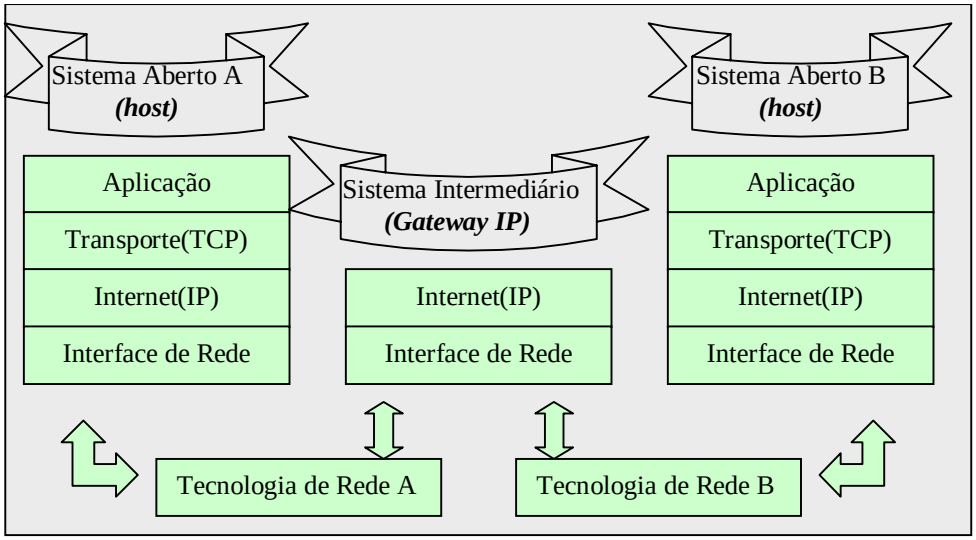


Figura A.1: Arquitetura TCP/IP.

- ✓ **Camada de Aplicação**: nesta camada, usuários invocam programas de aplicação que acessam serviços disponíveis através de uma *internet*. Uma aplicação interage

com um dos protocolos da camada de transporte para enviar e receber dados. Cada programa de aplicação escolhe o estilo de transporte necessário, o qual pode ser uma seqüência de mensagens individuais ou um *stream*[10]. O programa de aplicação passa os dados na forma requerida, pela camada de transporte, para entrega. Existem vários protocolos de aplicação que são utilizados pelos programas de usuários, ou até mesmo diretamente pelos próprios usuários (ex. HTTP, DNS, TELNET, FTP, etc..);

- ✓ **Camada de Transporte (TCP)**: esta camada provê comunicação fim-a-fim entre programas de aplicação, ou seja, ela não está interessada em quais ou quantas redes intermediárias as informações devem atravessar (a camada Internet resolve isso). Ela pode também realizar controle de fluxo, e congestionamento. Suas implementações geralmente oferecem duas alternativas (protocolos) à camada de aplicação:
  - ✓ *User Datagrama Protocol* (UDP) o qual provê um serviço de entrega de datagramas sem conexão e não confiável para transportar mensagens entre máquinas. Ele usa o IP para transportar mensagens, mas adiciona a habilidade de distinguir entre múltiplos destinos em um dado computador (*host*);
  - ✓ *Transmission Control Protocol* (TCP) provê uma conexão *full-duplex* entre duas máquinas, permitindo a troca de grandes volumes de dados (*streams*) eficientemente. O TCP implementa controle de fluxo (ex.: janela deslizante) e controle de congestionamento(ex.: *slow-start*);
- ✓ **Camada de Internet (IP)**: esta camada é encarregada de mover mensagens de uma máquina origem até o(s) destino(s). Para tal, ela define 3 importantes definições importantes: a unidade básica de informação chamada de datagrama; função de roteamento; conjunto de regras que personifica a entrega de pacotes sem conexão e não confiável.
- ✓ **Camada de Interface de Redes**: é responsável por aceitar datagramas IP e transmitir sobre uma rede específica. Uma interface de rede pode consistir de um *drive* de dispositivo (quando a rede é LAN – *Local Area Network* – para a qual a máquina está ligada diretamente) ou um subsistema complexo que usa seu próprio protocolo de enlace de dados (ex.: quando a rede consiste de comutadores de pacotes que se comunicam com *hosts* usando HDLC – *High Level Data Link Communication*).

---

[9] O *Open Systems Interconnection Reference Model (RM-OSI)* da *International Organization for Standarization (ISO)* define um padrão conceitual para interconexão de sistemas.
[10] *Stream*: transferência de uma grande quantidade de dados divididos em octetos.

A.2. Padronização Internet

O DARPA (*Defence Advanced Research Projects Agency*), patrocinou o desenvolvimento da arquitetura Internet TCP/IP. Hoje o corpo técnico do Comitê IAB (*Internet Activity Board*), coordena o desenvolvimento dos protocolos da arquitetura Internet TCP/IP. O IAB é formado por pesquisadores seniores, tendo a maioria deles projetado e implementado os protocolos da arquitetura Internet. Qualquer pessoa pode colaborar projetando, documentando, implementando e testando um protocolo, que possivelmente, poderá ser utilizado na Internet [Comer, 95].

Muitos dos protocolos que fazem parte da arquitetura TCP/IP foram padronizados ou estão em processo de padronização. Por um acordo universal uma organização conhecida como *Internet Architecture Board* (IAB) é responsável pelo desenvolvimento e publicação destes padrões, os quais são publicados em uma série de documentos chamados de *Request For Comments* (RFCs) [Comer, 95].

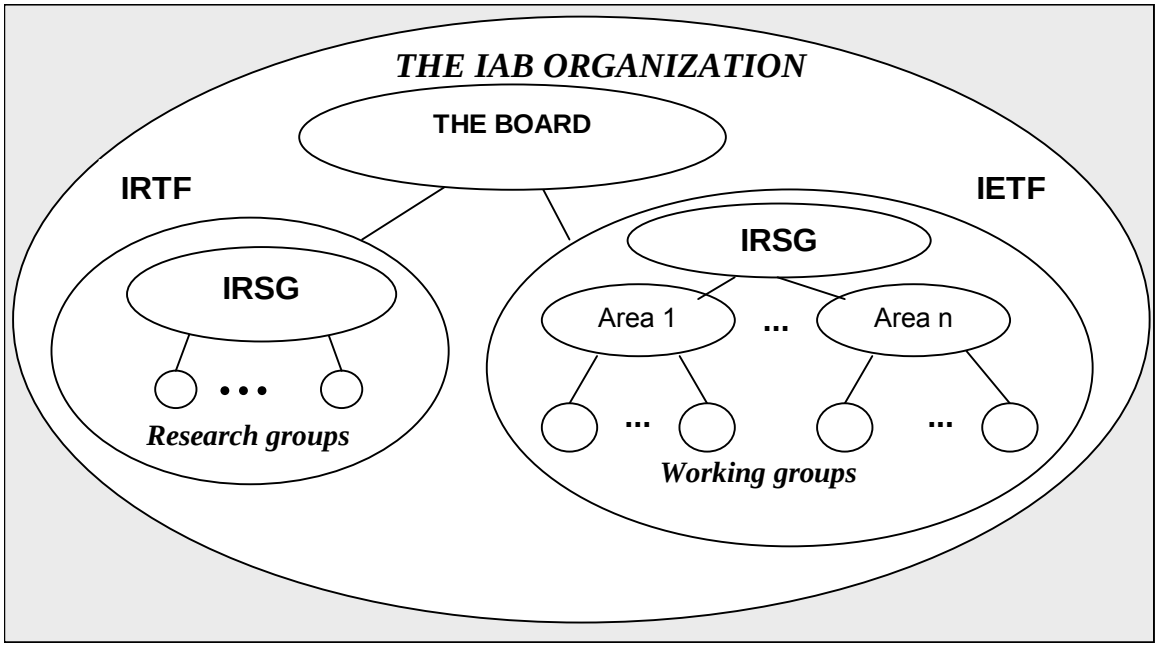


Figura A.3: A estrutura da IAB após reorganização em 1.989 [Comer 95].

A IAB é dividida principalmente em duas grandes forças: a IETF (*Internet Engineering Task Force*) e a IRTF (*Internet Research Task Force*). Estas Forças são formadas por grupos de trabalhos, onde qualquer pessoa pode participar destes grupos.

A IETF é responsável por publicar as RFCs (http://www.ietf.org), onde estão anotados trabalhos da comunidade de pesquisa e desenvolvimento da Internet. Os critérios de padronização da IAB dão ênfase à experiência operacional. O processo de padronização Internet é definido na RFC 2026 (*The Internet Standard Process Revision*).

Apêndice B

B. Protocolos de Roteamento Dinâmico

Apresenta-se neste apêndice os principais protocolos de roteamento dinâmico da tecnologia TCP/IP, segundo uma classificação relacionada com a forma de comunicação: *unicast* e *multicast*.

B.1. Protocolos de Roteamento Unicast

B.1.1. Routing Information Protocol - RIP

O RIP foi desenvolvido pela *Xerox Corporation* como parte da arquitetura de redes *Xerox Network Systems*(XNS), e também foi usado como base para o programa "roted", incluído UNIX distribuído pela *Berkeley*. Foi portado para o TCP/IP, no inicio da década de 80, quando surgiram as primeiras LANs. Hoje o RIP esta disponível na maioria das versões do sistema operacional UNIX e é suportado por todos os fabricantes de roteadores. E consequentemente, é o protocolo de roteamento mais utilizado na Internet, na atualidade.

O RIP utiliza um algoritmo do tipo Vetor de Distância, que tem sido usado para computações de roteamento em redes de computadores desde os primeiros dias da ARPANET. Suas principais características são [Hendrick, 88]:

- ✓ Projetado como um protocolo intra-domínio(IGP);
- ✓ Utiliza uma única métrica, a qual representa o nº de saltos da origem até o destino;

- ✓ Sua extensão esta limitada a 15 saltos, onde a métrica de uma rota com valor igual a 16 representa o infinito (rota inatingível);
- ✓ Período entre atualizações regulares (*update time*) é de 30 segundos;
- ✓ Tempo admitido sem receber a tabela de um vizinho (*invalid time*) é de 180 segundos, e quando é extrapolado este tempo considera-se a queda do enlace;
- ✓ As tabelas de roteamento guardam uma única rota para cada nó do sistema autônomo. As informações das tabelas são: Endereço de destino, Endereço do próximo nó, Interface a ser utilizada, Métrica, *Flags* e *Timers* que controlam tempos de atualização;
- ✓ Implementa os seguintes mecanismos de estabilidade para solucionar os problemas relacionados ao algoritmo vetor de distância citados no capitulo anterior: *Hop-count limit*, *Hold-down* (apenas no RIPv2), *Split horizon, Poison reverse updates, Triggerede updates*;

A versão mais atual deste protocolo é o RIPv2, a qual provê extensões para o formato das mensagens, possibilitando o tratamento de considerações importantes tais como autenticação, segurança, endereço *multicast* entre outros, suporte a CIDR[11]. A RFC 2080 especifica a versão do RIP para o IPv6, o qual especifica pouquíssimas modificações [Malkin, 97].

Os grandes benefícios do RIP é a facilidade de configuração e a necessidade de pouco poder de computação e capacidade de memória em roteadores e computadores.

B.1.2. Open Shortest Path First - OSPF

O OSPF é um protocolo relativamente novo, projetado pelo OSPF *Working Group* do IETF. Atualmente ele está na versão 2 [Moy, 97]. O OSPF é um protocolo "*Shortest Path First*", o qual baseia suas decisões nos registros de estado de enlace, que são atualizados dinamicamente. Seu nome vem do algoritmo "*Shortest Path First*" desenvolvido por [Dijkstra, 59]. Algoritmo este é usado pelos nós da rede para computar o melhor caminho. O OSPF é um protocolo de roteamento com um conjunto de opções e características complexas, contudo nem todas as caraterísticas estão disponíveis em todas as implementações. As principais características do OSPF são [Wessman, 96]:

- ✓ É um protocolo da classe de algoritmos Estado de Enlace (*Link-State Algoritms)*
- ✓ Oferece pequeno período de convergência (prevenindo *loops* de pacotes);
- ✓ Suporta múltiplas métricas;
- ✓ Suporta múltiplos caminhos para um destino. Quando dois ou mais caminhos tiverem com suas respectivas métricas dentro de uma faixa de valores pré-

[11] *Classless Interdomain Routing* (CIDR) define uma distribuição de endereços e estratégias de agregação (prefixos) designadas a minimizar o tamanho das tabelas de roteamento.

determinada, o fluxo pode ser dividido (proporcionalmente aos valores das métricas) entre estes caminhos;

- ✓ Usa uma representação separada para rotas externas (isto é útil para usar em conjunto com EGPs).
- ✓ Cada roteador do sistema autônomo OSPF mantém um mapa (base de dados) completo da topologia da rede, e executam computações locais para encontrar as melhores rotas baseadas neste mapa interno.
- ✓ Escalabilidade: OSPF é especificamente projetado para operar com grandes redes. Ele não impõe restrições ao nº de saltos e permite que seu domínio seja subdividido para facilitar o gerenciamento.
- ✓ Suporta *subnetting* completa: OSPF pode completamente suportar *subnetting* e sub-redes não contínuas;
- ✓ *Hello packets*: OSPF usa pequenos "*Hello packets*" para descobrir e manter relacionamentos com os vizinhos sem precisar transferir grandes tabelas. Em redes estáveis, grandes atualizações ocorrem em intervalos de 30 minutos;
- ✓ Roteamento por tipo de serviço (*TOS Routing*): pode optar por rotas diferentes para pacotes de aplicações distintas. Baseia-se no campo *Type of Service* (TOS), o qual oferece várias opções, tais como: atraso mínimo, *throughput* máximo, confiabilidade máxima, mínimo custo monetário, serviço normal. Exemplo: um acesso FTP poderia ser feito através de um enlace de satélite, enquanto um terminal I/O remoto utilizaria um enlace com menor retardo;
- ✓ O protocolo ainda especifica que todos os anúncios entre roteadores podem ser autenticados.

O OSPF em sua especificação inclui 3 protocolos diferentes [Wessman,96]:

- ✓ **Protocolo "HELLO"**: verifica quais enlaces estão operacionais, e também é usado para eleger um "Roteador Designado" e um "Roteador Designado Reserva";
- ✓ **Protocolo "EXCHANGE"** é usado para sincronizar as bases de dados entre dois nós, onde um nó age como "mestre" o outro como "escravo";
- ✓ **Protocolo "FLOODING"** é usado para propagar mudanças no estado de enlace para todos os outros nós da rede.

Quando uma rede é incorporada ao sistema autônomo OSPF, cada nó deve construir sua base de dados da topologia da rede. Para simplificar este passo e para limitar o nº de trocas de informações necessárias, os nós OSPF "elegem" um dos nós para agir como Roteador Designado, e um Roteador Designado Reserva (para agir caso o primeiro falhar). O Roteador designado também age como coordenador do envio de mensagens "*Flooding*" (ex. Mensagens sobre mudanças na topologia).

Todos os roteadores rodam exatamente o mesmo algoritmo, em paralelo. A partir do mapa de estado de enlace, cada roteador constrói uma árvore do melhor caminho, sendo ele próprio a raiz. Esta árvore contém o melhor caminho para cada destino no sistema autônomo. Externamente informações de roteamento aparecem na árvore como folhas [Moy, 97].

O OSPF é o protocolo recomendado pela IETF para ser usado como IGP sobre o IPv4, e também tem uma versão para ser usado no IPv6 especificada na RFC 2740. Algumas mudanças foram realizadas no OSPF atuar sobre o IPv6 (ex. acomodar o novo formato do endereço), mas a maioria dos algoritmos foram preservados [Coltun, 99].

B.1.3. Border Gateway Protocol -BGP

O BGPv4 é um protocolo EGP[12] , consistindo-se no atual padrão da Internet. O BGP, que é definido pelas RFCs(11771, 1772, 1773, 1774, 1657), é essencialmente um algoritmo vetor de distância, mas com várias distorções adicionadas.

O BGP utiliza a camada de transporte TCP, em vez da UDP utilizado por outros protocolos (ex. IDRP). Isto simplifica a máquina de estados do protocolo, já que ele pode confiar na camada TCP para entrega de mensagens. Outro fator favorável é à diminuição da carga da rede, já que uma camada de transporte segura possibilita atualizações incrementais, em vez da solução tradicional de copiar a base de dados inteira. Depois do estágio de inicialização, o BGP consome extremamente pouca largura de banda [Wessman, 96].

A unidade básica das informações de roteamento do BGP é o ***BGP path***. Os *Path*s são "etiquetados" com vários ***path attributes,*** dos quais os mais importantes são **AS_PATH** e **NEXT_HOP**. Os roteadores anunciam a rota completa entre a origem e o destino. O BGPv4 suporta a agregação de tabelas de roteamento requerida pelo CIDR. Ele insere, no *path attributes,* uma lista completa dos sistemas autônomos (conjunto de prefixos CIDR) que serão usados para o transporte dos dados. Isto permite a implementação de políticas de roteamento e detecção completa de *loop* a nível de sistema autônomo [Wessman,96].

Cada vez que uma mensagem BGP *path* passa de um sistema autônomo para outro, o atributo HEXT_HOP é atribuído com o endereço IP do roteador de borda, o que não ocorre quando a mensagem é passada entre roteadores do mesmo sistema autônomo. Consequentemente, o NEXT_HOP é sempre o endereço do primeiro roteador do próximo sistema autônomo. O IGP do sistema autônomo é responsável por computar uma rota interior para alcançar o BGP NEXT_HOP. Isto leva a distinção entre sessão *Internal* BGP (IBGP) e sessão *External* BGP (EBGP). Em resumo uma sessão IBGP é realizada entre roteadores do mesmo sistema autônomo e não tem capacidade de mudar o valor do atributo NEXT_HOP, e uma sessão EBGP se dá entre sistemas autônomos e cada vez que ocorre o valor do NEXT_HOP é alterado.

[12] EGP (Exterior Gateway Protocol) também é o nome do antecessor do BGP, que leva o mesmo nome da classe de protocolos responsável pelo roteamento inter-domínio(EGP), a qual eles fazem parte protocolo

B.2. Protocolos de Roteamento Multicast

B.2.1. Distance Vector Multicast Routing Protocol (DVMRP)

O DVMRP é o primeiro protocolo desenvolvido para suportar roteamento *multicast*. A primeira versão deste protocolo é descrito na RFC 1075, e atualmente existe um *Draft* [Pusateri, 99], o qual descreve sua terceira versão.

Este protocolo se tornou popular após a implementação do software *mrouted*, utilizado em grande escala no MBONE[13]. O *mrouted* foi desenvolvido tomando como base o DVMRP, com a introdução de "podas" nos ramos inúteis das árvores inicialmente construídas para um par (Grupo, Fonte), de acordo com o algoritmo *Reverse Path Multicast* (RPM) [Kronbauer, 98]. Está fora do escopo deste documento as redes que contém roteadores que não suportam *multicast*, ou seja, que necessitam de *Tunneling*.

O DVMRP constrói uma *spanning tree* para cada grupo *multicast* e seus receptores. A *spanning tree* representa o menor caminho entre a fonte (raiz) e os componentes do grupo (folhas). O menor caminho é calculado com base na métrica DVMRP, a qual é o número de saltos entre a fonte e cada destino. A construção da árvore é realizada sob demanda usando uma técnica *"broadcast and prune"*, ou seja, quando uma fonte inicia a transmissão de mensagens para um grupo *multicast* [Jonhson, 97].

O Protocolo DVMRP assume inicialmente que cada *host* na rede faz parte do grupo *multicast*. O roteador designado na sub-rede fonte (ex. o roteador selecionado para tratar o roteamento para todos os *hosts* em sua sub-rede), começa pela transmissão de uma mensagem *multicast* para todos roteadores adjacentes. Cada um destes roteadores então repassam a mensagem seletivamente para os roteadores *downstream*, até que a mensagem é eventualmente passada para todos membros do grupo *multicast*.

Roteadores DVMRP vizinhos são descobertos dinamicamente pelo envio de mensagens de descoberta de vizinhos (*Neighbor Probe Messages*) em interfaces de rede local com capacidade *multicast* e interfaces com pseudo túnel (*Tunneling*). Estas mensagens são enviadas periodicamente para o endereço de grupo *multicast "All-DVMRP-Routers"*. O TTL destas mensagens deve ser 1. Cada mensagem contem a lista de roteadores vizinhos DVMRP dos quais ele recebeu mensagens *Neighbor Probe* por aquela interface, tendo com isso estabelecido uma adjacência com este roteador vizinho.

Quando um datagrama IP *Multicast* é recebido por um roteador rodando o DVMRP, ele procura o endereço da rede fonte na tabela de roteamento DVMRP. A interface na qual a melhor

[13] MBONE – *Backbone Multicast* – é uma implementação de protocolos de roteamento *multicast* na Internet, e seus principais objetivos são: promover a pesquisa para o desenvolvimento de protocolos *multicast*; desenvolvimento de novas aplicações multimídia que possam utilizar as facilidades *multicast*, e viabilizar conectividades entre domínios diferentes, separados por domínios que não implementam o *multicast(Tunneling)*.

rota (até a fonte) do datagrama recebido é chamada de interface *upstream*. Se o datagrama chegar pela interface *upstream* correta, então ele será repassado para uma ou mais interfaces *downstream*. Se o datagrama não chegar pela interface *upstream* prevista, ele será descartado. [Pusateri, 99].

B.2.2. MOSPF – Multicast Extension OSPF

O MOSPF permite que seja realizado roteamento *multicast* em um domínio de roteamento OSPF *unicast*. Este protocolo define um novo tipo de *Link Statment Advertisement* - LSA (*Group-Membership-LSA*) para base de dados de estado de enlace e adiciona cálculos para os caminhos dos datagramas *multicast*. O cálculo do melhor caminho, realizado pelo MOSPF, utiliza a métrica de enlace do OSPF, e utiliza um algoritmo do tipo árvore baseado na fonte.

Hosts anunciam sua junção a um grupo enviando uma mensagem *Host-Membership-Report*, e Roteadores verificam a existência de membros de um grupo enviando a mensagem H*ost-Membership-query*.

O *Group-Mempbership-LSA* é uma mensagem IGMP enviada pelo "Roteador Designado" de um segmento de rede para todos os outros roteadores do domínio de roteamento *multicast*.

Os roteadores MOSPF calculam os caminhos *multicast* entre uma fonte *multicast* e os membros de grupo destino. A combinação entre fonte e grupo de destino pode ser representada pelo par (F,G)

O MOSPF tem diferentes caminhos para cada combinação de rede fonte e grupo *multicast* destino. A carga de cálculos que os roteadores MOSPF agüentam é reduzida pelo fato de que os cálculos de roteamento *multicast* são executados sob demanda, ou seja, quando um roteador MOSPF recebe um datagrama de uma fonte *multicast*.

Esta conduta é realizada conforme segue [Moy, 97]:

- ✓ 1º) ele realiza o número de cálculos de roteamento no decorrer do tempo
- ✓ 2º) Um roteador MOSPF não precisa calcular o caminho entre uma dada combinação de (F,G) a não ser que ele estiver no caminho entre a fonte (F) e um ou mais membros do grupo (G).

Caminhos *multicast* são recalculados quando há alterações na rede, tais como:

- ✓ Roteadores e enlaces entram ou saem de funcionamento na rede;
- ✓ *Host* juntam-se ou deixam um grupo *multicast*;

O roteador MOSPF calcula o caminho entre uma fonte de um grupo e todos os membros destinos deste grupo. Entretanto o roteador não precisa armazenar o caminho completo. Somente a posição do roteador relacionada ao caminho é importante. Esta posição é armazenada em uma entrada no *multicast forwarding cache* (MFC). Cada caminho destino entre uma fonte e seu

respectivo grupo de destinos representa uma entrada no MFC. Cada entrada MFC do roteador contém as seguintes informações:

- ✓ O ID do roteador ou da rede do qual o roteador deve receber os datagramas do grupo correspondente. Isto serve para assegurar que somente uma copia do datagrama é recebida em algum segmento em particular;
- ✓ A Interface de saída na qual o roteador deve enviar os datagramas, juntamente com o número de saltos para alcançar os membros do grupo mais próximo. Isto será representado pelo TTL do pacote a ser enviado.

Um roteador pode limpar uma ou mais de suas entradas no MFC em qualquer momento, pois estas entradas serão reconstruídas no instante que receber um datagrama da combinação. Entretanto, certas entradas no MFC devem ser eliminadas quando houver alguma modificação nas condições da rede.

Ao ocorrer uma modificação no domínio de roteamento *multicast*, certas entradas no MFC devem ser eliminadas. Entretanto, não são reconstruídas automaticamente. As entradas serão reconstruídas de acordo com a chegada dos datagramas da combinação (F,G) afetada pela mudança, da mesma forma que foram construídas na primeira vez.

O roteador sabe que ocorreu mudanças quando recebe LSAs atualizados. O tipo de LSA determina a natureza e magnitude das mudanças, por exemplo:

- ✓ *Router-LSA* ou *Network-LSA*: indica que um roteador ou enlace caiu ou foi restaurado, ou apenas pode ter mudado o custo (métrica) de um enlace. – Neste caso é impossível saber precisamente qual caminho foi afetado, portanto todas as entradas do MFC devem ser *eliminadas*;
- ✓ *Group-Membership-LSA*: indica que ouve mudanças nos membros de um grupo *multicast*. Neste caso somente as entradas correspondentes a este grupo devem ser *eliminadas.*

Cálculo do roteamento *multicast* MOSPF é semelhante ao cálculo do roteamento do OSPF, entretanto, tem algumas diferenças [Moy, 97]:

Os roteadores *unicast* baseiam-se puramente no destino do datagrama ao calcular as entradas de suas tabelas de rotas. O próprio roteador é a raiz da árvore de menor caminho; Os roteadores *multicast* devem se preocupar tanto com a fonte quanto com os destinos dos datagramas ao calcular as entradas de sua tabela de rotas (*Multicast Forwarding Cache*). Isto força o cálculo da árvore de menor caminho com base na fonte (a raiz da árvore e a fonte). Uma conseqüência desta diferença é que os cálculos *unicast* são sempre específicos para cada roteador, e todos os roteadores *multicast* sempre executam os mesmos cálculos para um determinado datagrama *mulicast*. Segunda diferença quanto ao tratamento de "*Stub-Network*", no MOSPF os membros de grupo de uma *stub-Network* são agregados e anunciados pelo roteador da *stub network*. A terceira diferença, cálculos OSPF mantém caminhos redundantes para aplicar

suas características de balanceamento de tráfego, mas o MOSPF elimina caminhos redundantes para evitar replicações de pacotes, onde caminhos alternativos são podados da árvore de melhor caminho. Quando há mais de um caminho com o mesmo custo (métrica) o MOSPF introduz o cálculo "*Tie Breaker*" (o caminho cujo roteador (ou rede) do salto prévio tiver o maior endereço é escolhido).

O algoritmo *Dijkstra* calcula rotas para todos os destinos de uma vez. Contudo, o MOSPF está interessado apenas nos caminhos de uma fonte para todos os destinos de um grupo específico. Portanto a parte final dos cálculos de roteamento do MOSPF é a rotina de poda dos caminhos que levem aos não membros do grupo.

B.2.3. Outros Protocolos Multicast

O *Protocol Independent Multicast* (PIM) é um protocolo de roteamento *multicast*, cuja segunda versão está em desenvolvimento, que utiliza informações sobre rotas ponto-a-ponto, obtidas através de um protocolo de roteamento *unicast* arbitrário. O PIM também distingue a forma de roteamento para grupos *multicast* densos (PIM – *Dense Mode*) e esparsos (PIM – *Sparse Mode*). A especificações destes protocolos estão descritas em e [Helmy, 98] e [Helmy, 99], respectivamente.

O *Core Based Tree* (CBT) é um protocolo concebido com a intenção principal de aumentar a escalabilidade apresentada nos protocolos que constróem árvores de roteamento para cada fonte *multicast*. A abordagem consiste em implementar uma única árvore de roteamento para cada grupo, que será utilizada por todas as fontes do grupo [Kronbauer, 98]. O CBT se encontra na sua segunda versão e sua especificação é definida em [Ballardie, 97].

O *Border Gateway Multicast Protocol* (BGMP) é um protocolo designado para o roteamento *multicast* entre sistemas autônomos (domínio). Este protocolo constrói árvores de caminhos compartilhadas para ativar grupos *multicast*, e requer que cada grupo *multicast* seja associado a um único "domínio raiz", em vez de roteador raiz como o CBT, por exemplo [Thaler, 00]. Este protocolo está em fase de especificação.

Apêndice C

C. Algoritmos de Roteamento

O algoritmo de roteamento é a parte do software da camada de rede responsável por decidir sobre qual linha de saída (enlace) um pacote, que chega a determinado nó, deve ser transmitido. Contudo, independente de um algoritmo específico há algumas propriedades desejáveis para assegurar a eficácia dos mesmos [Tanembaun, 97]:

- ✓ Correção: calcular corretamente as melhores rotas, tendo o cuidado para não informar, erroneamente, rotas inexistentes;
- ✓ Simplicidade: deve agir com eficiência sem sobrecarregar a máquina com cálculos complexos. O administrador da rede deve entender o funcionamento do algoritmo;
- ✓ Resistência: o algoritmo deve estar apto a funcionar continuamente, suportando e agindo corretamente sob várias falhas de hardware e software que os nós venham a sofrer. Falhas estas, causam mudanças na topologia e no tráfego;
- ✓ Estabilidade: os nós da rede devem convergir rapidamente. Convergir diz-se quando todos os nós ficam com suas tabelas de roteamento completas e atualizadas com as rotas ótimas;
- ✓ Consideração com o usuário e eficiência global: estas duas propriedades muitas vezes são contraditórias, pois, muitas vezes, para melhorar a eficiência global deve-se sacrificar a comunicação entre alguns usuários. Contudo, esses usuários podem não estar de acordo com esta decisão. Deve-se decidir o que se busca otimizar para tomar decisões ponderadas.

Os algoritmos de roteamento podem ser agrupado em duas classes [Soares, 95]:

- ✓ Estáticos (não adaptativos): as rotas são computadas com antecedência e alocadas na tabela de roteamento de cada nó que, uma vez criada, não é mais alterada. Os nós recebem suas tabelas no momento da inicialização da rede. Se o tráfego da rede não for regular e bem conhecido, haverá uma má utilização dos meios de comunicação;
- ✓ Dinâmicos (adaptativos): alteram dinamicamente suas tabelas de roteamento, baseando-se nas mudanças na topologia e no tráfego corrente. Este tipo de encaminhamento pode ser subdividido de acordo com a forma de atualização das tabelas, tais como, centralizados, isolados ou distribuídos.

A seguir, são descritos dois dos algoritmos de roteamento mais conhecidos: Vetor Distância e Estado de Enlace. Mais detalhes sobre algoritmos de roteamento encontram-se em [Tanembaun, 97]).

C.1. Algoritmo Vetor de Distância (Distance Vector Algorithm)

A primeira descrição, conhecida, desta classe de algoritmo é [Ford, 62] e por este motivo este algoritmo também é conhecido como Ford-Fulkerson, contudo o termo Bellman-Ford também é usado devido ao fato de que sua formulação foi baseada na equação de Bellman, a base da programação dinâmica [Hendrick, 88]. Segue um resumo de seu funcionamento descrito em [Schimitt, 96]:

- ✓ Em intervalos de tempo regulares cada nó da rede envia toda a sua tabela de roteamento para, e somente para, os seus vizinhos;
- ✓ Após algum tempo os diversos nós da rede convergem, ou seja ficam com as suas tabelas de roteamento completas (uma entrada para cada destino) e atualizadas;
- ✓ As tabelas de roteamento contém, pelo menos, o endereço destino, a métrica, e o próximo nó para onde a mensagem deve ser enviada;
- ✓ Exige poucos recursos de memória e de processamento dos nós da rede;

Contudo, apesar da convergência mais lenta, existem alguns problemas relacionados com o algoritmo Vetor de Distância [Soares, 97b]:

- ✓ ***Bouncing Effect:*** este problema corresponde à formação de laços durante a fase de convergência do algoritmo. As mensagens enviadas ficam rebatendo de um roteador para outro e nunca chegam ao seu destino, sendo eliminadas por causa do TTL (Time To Live);
- ✓ ***Count to infinity:*** corresponde à impossibilidade do algoritmo convergir em algumas situações de queda de enlaces. A rota nunca é marcada como inatingível e os pacotes ficam novamente rebatendo de um nó para outro.

A seguir apresentam-se várias soluções encontradas para resolução total ou parcial dos problemas citados acima:

- ✓ ***Hop-count Limit:*** o infinito é representado por um número. Este número deve ser escolhido de maneira tal que o algoritmo não demore muito para convergir, mas também que a distância máxima entre dois nós da rede não fique muito limitada;
- ✓ ***Split Horizon:*** as informações de roteamento não devem ser enviadas de volta para a máquina que as originou. Este mecanismo soluciona o problema de laço entre 2 nós, mas não soluciona o problema de laços entre mais de 2 nós;
- ✓ ***Hold-down:*** determina que quando uma rota torna-se inatingível ela só possa ser restabelecida após algum tempo. Isto impede que uma rota que caia volte precipitadamente. Este período deve ser maior do que o tempo necessário para que uma mensagem se propague por toda a rede;
- ✓ ***Triggered Updates (atualizações disparadas):*** na ocorrência de uma modificação na tabela de roteamento, imediatamente mensagens de atualização são disparadas. Não sendo aguardado o intervalo normal de transmissão das mensagens de atualização.

C.2. Algoritmo Estado do Enlace (Link State Algorithm).

Esta classe de algoritmo utiliza mais de um algoritmo de roteamento, tais como o algoritmo de roteamento pelo caminho mais curto desenvolvido por [Dijkstra, 59] e o algoritmo Dilúvio (*Flood*), entre outros. Segue seu funcionamento básico:

- ✓ Descobre quem são os vizinhos e qual o estado dos enlaces deles;
- ✓ Mede os custos associados aos diversos enlaces que possui;
- ✓ Transmite as informações sobre os enlaces para todos os nó da rede;
- ✓ Recebe o estado de todos os enlaces da rede;
- ✓ Constrói um mapa completo da rede;
- ✓ Constrói o melhor caminhos para cada roteador da rede utilizando o algoritmo de Dijkstra;
- ✓ Cada roteador constrói uma árvore de melhor caminho, sendo ele próprio a raiz.

As principais características deste algoritmo são [Shimitt, 96]:

- ✓ As atualizações ocorrem somente quando há uma modificação da rede;
- ✓ As atualizações são enviadas para todos os roteadores da rede;
- ✓ Para realizar o envio da atualização para toda a rede, existe um protocolo que "inunda" a rede com a mensagem, chamado de *flooding protocol*;
- ✓ Quando dois nós se "descobrem", durante a inicialização de um roteador, eles trocam mensagens entre si para sincronizarem os seus bancos de dados. Depois de os bancos de dados estarem iguais, a nova configuração é comunicada ao restante da rede;

- ✓ Os bancos de dados topológicos de todos os roteadores devem estar sempre iguais após a convergência do algoritmo;
- ✓ Apresentam uma rápida convergência;
- ✓ É bastante complexo e requer grande poder de processamento e ocupa bastante memória dos nós da rede.

Apêndice D

D. Acrônimos

ATM	*Asynchronous Transfer Mode*
BGMP	*Border Gateway Multicast Protocol*
BGP	*Border Gateway Protocol*
CBT	*Core Border Tree*
CCR	Centro de Controle de Roteamento
CIDR	*Classless Inter Domain Routing*
DAO	*Data Access Objects*
DARPA	*Defence Advanced Research Projects Agency*
DHCP	*Dynamic Host Configuration Protocol*
DVMRP	*Distance Vector Multicast Routing Protocol*
EBGP	*Exterior Border Gateway Protocol*
EGP	*Exterior Gateway Protocol*
FTP	*File Transfer Protocol*
HDLC	*High-Level Data Link Control*
I/O	*Input/Output*
IAB	*Internet Activity Board*
ICMP	*Internet Control Message Protocol*
IDRP	*Inter Domain Routing Protocol*
IETF	*Internet Engineering Task Force*
IGMP	*Internet Group Multicast Protocol*
IGP	*Interior Gateway Protocol*
IP	*Internet Protocol*
IRTF	*Internet Research Task Force*
LAN	*Local Area Network*
LSA	*Link Statment Advertisement*
MAC	*Medium Access Control*
MBONE	*Multicast Backbone*
MFC	*Microsoft Fundation Class*
MFC	*Multicast Forwarding Cache*

MOSPF	*Multicast Open Shortest Path First*
MTU	*Minimum Time Unity*
OOS	*Oriented Object Simulator*
OSPF	*Open Shortest Path First*
PC	*Personal Computer*
PIM	*Protocol Independent Multicast*
PIM-DM	*Protocol Independent Multicast - Dense Mode*
PIM-SM	*Protocol Independent Multicast - Sparse Mode*
RFC	*Request For Comments*
RIP	*Routing Internet Protocol*
RM-OSI	*Reference Model – Open Systems Interconnection*
RPM	*Reverse Path Multicast*
RSVP	*Reservation Protocol*
SAVAD	Sistema Especialista para Avaliação de Desempenho de Modelos de Redes de Filas
SNAcP	*SubNetwork Access Protocol*
SNDcP	*SubNetwork Dependent Protocol*
SNIcP	*SubNetwork Independent Protocol*
SQL	*Search Query Language*
TCP	*Transmission Control Protocol*
TTL	*Time To Live*
UDP	*User Datagrama Protocol*
UFPB	*Universidade Federal da Paraíba*
UML	*Unified Modeling Language*
XNS	*Xerox Network Systems*
VBA	*Visual Basic Application*