PROJETO FINAL DE GRADUAÇÃO

PROPOSTA DE SIMULAÇÃO E ANÁLISE DE APLICAÇÕES MULTIMÍDIA EM REDES 4IN6 E 6IN4 EM AMBIENTE DE REDE VIRTUALIZADO

Lucas Rodrigues de Freitas Ribeiro

Paulo Henrique Alves Torres Leal

Brasília, Dezembro de 2016

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

PROJETO FINAL DE GRADUAÇÃO

PROPOSTA DE SIMULAÇÃO E ANÁLISE DE APLICAÇÕES MULTIMÍDIA EM REDES 4IN6 E 6IN4 EM AMBIENTE DE REDE VIRTUALIZADO

Lucas Rodrigues de Freitas Ribeiro

Paulo Henrique Alves Torres Leal

Projeto Final de Graduação submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro de Redes de Comunicação.

Banca Examinadora

Prof. Georges Daniel Amvame Nze, Dr., UnB _____
Orientador

Prof. Robson de Oliveira Albuquerque, Dr., ENE/UnB _____
Examinador interno

Diego Martins de Oliveira, Esp., IFB _____
Examinador externo

FICHA CATALOGRÁFICA

LEAL, PAULO HENRIQUE ALVES TORRES ; RIBEIRO, LUCAS RODRIGUES FREITAS
PROPOSTA DE SIMULAÇÃO E ANÁLISE DE APLICAÇÕES MULTIMÍDIA EM REDES 4IN6 E 6IN4 EM AMBIENTE DE REDE VIRTUALIZADO [Distrito Federal] 2016.
xvi, 62 p., 210 x 297 mm (ENE/FT/UnB, Engenheiro, Engenharia Elétrica, 2016).
Projeto Final de Graduação - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Redes virtualizadas
2. Linux
3. Redes 4in6 e 6in4
4. Virtualização

I. ENE/FT/UnB
II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

LEAL, P. H. A. T. ; RIBEIRO, L. R. F. (2016). *PROPOSTA DE SIMULAÇÃO E ANÁLISE DE APLICAÇÕES MULTIMÍDIA EM REDES 4IN6 E 6IN4 EM AMBIENTE DE REDE VIRTUALIZADO*. Projeto Final de Graduação, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 62 p.

CESSÃO DE DIREITOS

AUTORES: Lucas Rodrigues de Freitas Ribeiro, Paulo Henrique Alves Torres Leal
TÍTULO: PROPOSTA DE SIMULAÇÃO E ANÁLISE DE APLICAÇÕES MULTIMÍDIA EM REDES 4IN6 E 6IN4 EM AMBIENTE DE REDE VIRTUALIZADO.
GRAU: Engenheiro de Redes de Comunicação ANO: 2016

É concedida à Universidade de Brasília permissão para reproduzir cópias deste Projeto Final de Graduação e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. Os autores reservam outros direitos de publicação e nenhuma parte desse Projeto Final de Graduação pode ser reproduzida sem autorização por escrito dos autores.

Lucas Rodrigues de Freitas Ribeiro
Paulo Henrique Alves Torres Leal
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

DEDICATÓRIAS

Àqueles que persistem com fé em Deus.

Lucas Rodrigues de Freitas Ribeiro

A qualquer um que um dia tenha acreditado no sucesso desta jornada.

Paulo Henrique Alves Torres Leal

AGRADECIMENTOS

Agradeço primeiro a Deus, pelo sopro de vida e esperança; aos meus pais, em especial à minha mãe Maria que trabalhou dia e noite pra que eu tivesse condições de estudar, ao meu pai por sempre acreditar em mim e à minha irmã por ter me dado um novo propósito na vida; aos meus familiares que juntos ou distantes sempre foram exemplo de fé; aos amigos de faculdade que sempre me socorreram e me ajudaram em minhas dificuldades, em especial ao Paulo e Willian, que são exemplos de superação e vitória; aos meus amigos de infância, que mesmo quando distantes sempre me ajudam, em especial ao Anderson por ter me ensinado novamente o que é ter fé, e ao Afonso pela amizade perene e exemplo de luta; ao meu orientador por ter confiado essa missão e sempre ser solícito na passagem de conhecimento; por fim, a todos os que compartilharam conhecimento e fizeram parte dessa árdua e divertida caminhada possível.

Lucas Rodrigues de Freitas Ribeiro

Agradeço primeiramente a Deus, por ter me proporcionado condições de conseguir tudo que tenho; aos meus pais e irmãos, pelo amor e apoio incondicional durante toda a vida; aos meus primos, tios e demais familiares, que me fazem perceber que o amor familiar é indispensável; aos amigos de faculdade, por tornarem esta longa caminhada muito mais tranquila; aos amigos de infância, que me acompanham até hoje e sempre acreditaram nas minhas conquistas; aos meus professores, que durante todos esses anos me ensinaram e me fizeram ter orgulho de fazer parte desta universidade, em especial ao meu orientador, pelas sugestões e conhecimento compartilhados; por fim, a qualquer um que um dia tenha acreditado que este sonho seria possível.

Paulo Henrique Alves Torres Leal

RESUMO

Atualmente, dada a característica diversificada das redes de computadores em relação aos protocolos de comunicação utilizados na camada de rede da arquitetura TCP/IP, torna-se necessária a utilização de mecanismos que permitam a coexistência destas. Mais especificamente, redes IP são divididas em redes IPv4 (protocolo IP, versão 4) e IPv6 (protocolo IP, versão 6), especificadas pelas RFC's 791 e 2460, respectivamente. Dessa forma, o tratamento adequado dos dados trafegados em redes que apresentam ambos os protocolos é de suma importância para a manutenção de aspectos essenciais da rede, como qualidade de serviço (QoS) e interoperabilidade de sistemas. As três principais técnicas utilizadas para comunicação entre redes IPv4 e IPv6 são o ***tunelamento***, a ***tradução*** e a ***pilha dupla (dual stack)***. Todas essas técnicas foram pensadas de forma a permitir a coexistência de ambos os tipos de rede, ou seja, são aplicadas apenas durante o processo de migração da rede por completo, e não de forma definitiva. O foco e objetivo deste projeto está na utilização do método de ***tradução*** utilizando roteadores que implementam o mecanismo de NAT64 (network address translation 6 to 4) e servidores DNS64 (tradução de nomes) para análise de tráfego em um ambiente de rede completamente virtualizado construído através de scripts Linux, utilizando-se ferramentas *open-source*, propostos também no escopo deste projeto. De forma geral, observa-se que o NAT64/DNS64 mostra-se como uma alternativa confiável e eficaz para ser utilizado em ambientes de redes diversos.

ABSTRACT

Currently, given the diversified characteristic of computers networks in relation to network communication protocols used in the layer 3 of the TCP/IP architecture, the use of some mechanisms becomes necessary to allow the coexistence of these. More specifically, IP networks are divided into IPv4 (Internet Protocol, version 4) and IPv6 (Internet Protocol, version 6), specified by RFC's 791 and 2460, respectively. Thus, the proper treatment of data trafficked in networks that have both protocols is very important for the maintenance of essential aspects of the network, such as quality of service (QoS) and interoperability of systems. The three main techniques used for communication between IPv4 and IPv6 networks are the ***Tunneling***, the ***translation*** and ***dual stack***. All of these techniques have been designed to allow coexistence of both network types, i.e., they are applied only during the network migration process altogether, and not permanently. The focus and goal of this project is the use of the ***translation*** method using routers that implement the NAT64 mechanism (network address translation 6 to 4) and DNS64 servers (translation of names) for traffic analysis in a completely virtualized network environment built with Linux scripts, using *open-source* tools, also proposed in the scope of this project. In general terms, it is observed that NAT64 / DNS64 mechanism is shown as a reliable and efficient alternative for use in differents network environments.

SUMÁRIO

1 Introdução 1
1.1 Motivação 1
1.2 Objetivo 1
1.2.1 Objetivos Específicos 2
1.3 Trabalhos Correlatados 2
1.4 Estrutura do trabalho 2

2 Fundamentação Teórica 4
2.1 Protocolos e Redes IP 4
2.1.1 Protocolo IPv4 4
2.1.2 Protocolo ICMPv4 9
2.1.3 Protocolo IPv6 10
2.1.4 Protocolo ICMPv6 13
2.1.5 Mecanismos de transição de redes IPv4 para IPv6 14
2.2 O Mecanismo de Tradução IPv4/IPv6 16
2.2.1 NAT - Network Address Translation 16
2.2.2 NAT64 - Network Address Translation 6to4 18
2.2.3 DNS64 - Domain Name System 6to4 24
2.3 Ambientes de rede virtualizados 27
2.4 Ambientes Linux 28
2.4.1 Ferramentas utilizadas 28

3 Metodologia experimental e estudo de caso 32
3.1 Proposta de ferramenta de configuração para redes virtualizadas 32
3.1.1 Configuração da interface de rede 32
3.1.2 Roteamento estático 33
3.1.3 Configuração via Quagga 34
3.1.4 NAT64 36
3.1.5 DNS64 37
3.1.6 Exemplo de caso de uso - NAT64 38
3.2 Topologias de teste 38
3.2.1 Topologia com endereçamento puramente IPv4 39
3.2.2 Topologia com endereçamento puramente IPv6 40
3.2.3 Topologia mista com NAT64 e DNS64 41
3.2.4 Topologia para testes de conexão com a Internet 42

3.3 Testes de desempenho de rede 43
3.3.1 Iperf 43
3.3.2 Troughput 44
3.3.3 Jitter 45
3.3.4 Latência 45

4 Resultados e Análise 46
4.1 Análise de verificação de conexão com NAT64 e DNS64 via *wireshark* 46
4.1.1 NAT64 46
4.1.2 DNS64 48
4.1.3 Teste de Conexão IPv6 com rede puramente IPv4 50
4.1.4 Teste de Conexão IPv6 com rede puramente IPv6 51
4.2 Análise de desempenho do NAT64 e DNS64 51
4.2.1 Teste de latência via ICMP Echo Request (PING) 51
4.2.2 Teste de throughput TCP 53
4.2.3 Teste de variação de atraso (Jitter) em aplicações UDP. 54

5 Conclusão e trabalhos futuros 59
5.1 Conclusões gerais 59
5.2 Trabalhos futuros 59

REFERÊNCIAS BIBLIOGRÁFICAS 61

LISTA DE FIGURAS

2.1 Ilustração geral de uma rede IP e das camadas TCP/IP 5
2.2 Formato do datagrama IP 5
2.3 Estrutura e exemplo de endereço IPv4 7
2.4 Estrutura de mensagem de erro ICMPv4 9
2.5 Estrutura do datagrama IPv6 11
2.6 Exemplo de endereço IPv6 e diferentes representações possíveis 12
2.7 Estrutura de mensagem IPv6/ICMPv6 14
2.8 Exemplo de arquitetura de rede com tunelamento 4in6 15
2.9 Exemplo de arquitetura de rede com tunelamento 6in4 15
2.10 Exemplo de arquitetura de rede com implementação de pilha dupla 16
2.11 Esquema de mapeamento endereço/porta 18
2.12 Processo de tradução 6to4 e 4to6 19
2.13 Representação IPv6 do endereço IPv4 22
2.14 Exemplo de sequência de funcionamento do NAT64 23
2.15 Exemplo de sequência de funcionamento do DNS64 25
2.16 Tipos de validação DNS64 com DNSSEC 26
2.17 Exemplo de sequência de funcionamento do DNS64 junto ao NAT64 para endereço traduzido 27
2.18 Arquitetura de rede virtual resumida 28
2.19 Processo de roteamento em ambiente GNU/Linux 29

3.1 Menu principal da ferramenta de configuração 32
3.2 Menus da configuração de interface de rede 33
3.3 Menus da configuração de roteamento estático 34
3.4 Menus da escolha do tipo de rota 34
3.5 Menus da configuração do módulo Quagga 35
3.6 Menus da configuração do roteamento OSPF via Quagga 35
3.7 Menus da configuração do NAT64 36
3.8 Menus da configuração do DNS64 37
3.9 Caso de uso para configuração NAT64 38
3.10 Topologia de testes de desempenho 39
3.11 Topologia de testes de conexão com a internet 43

4.1 Pacote IPv6 capturado no roteador R1 da topologia da figura 3.11 via wireshark antes da tradução NAT64 46
4.2 Pacote IPv4 capturado no roteador R1 da topologia da figura 3.11 via wireshark após a tradução NAT64 47

4.3 Requisição e resposta DNS para o domínio *www.nbc.com* capturado no roteador R1 da topologia da figura 3.11 via wireshark 48
4.4 Requisição e resposta DNS para o domínio *www.ene.unb.br* capturado no roteador R1 da topologia da figura 3.11 via wireshark 49
4.5 Requisições e respostas HTTP para o domínio *www.ene.unb.br* capturadas via wireshark no host H1 da topologia da figura 3.11 50
4.6 Handshake TCP para o domínio *www.v6.facebook.com* capturado via wireshark no host H1 da topologia da figura 3.11 51
4.7 Gráfico de média de latência referentes à figura da topologia 3.10 com endereçamento IPv4-Only, IPv6-Only e tradução NAT64 52
4.8 Gráfico de média de throughput referentes à figura da topologia 3.10 com endereçamento IPv4-Only, IPv6-Only e tradução NAT64 53
4.9 Gráfico de média na variação de atraso referentes à figura da topologia 3.10 com endereçamento IPv4-Only, IPv6-Only e tradução NAT64 em aplicação de áudio e vídeo de alta qualidade 55
4.10 Gráfico de média na variação de atraso referentes à figura da topologia 3.10 com endereçamento IPv4-Only, IPv6-Only e tradução NAT64 em aplicação de áudio e vídeo de alta qualidade 56
4.11 Gráfico de média na variação de atraso referentes à figura da topologia 3.10 com endereçamento IPv4-Only, IPv6-Only e tradução NAT64 em aplicação de áudio e vídeo de alta qualidade 57

LISTA DE TABELAS

1.1 Trabalhos Correlatados 2

2.1 Divisão de classes dos endereços IPv4 8
2.2 Divisão de endereços privados 9
2.3 Valores dos campos do cabeçalho IPv4 para mensagem ICMPv4 9
2.4 Valores do campo *type* para mensagem ICMPv4 e respectivos tipos de mensagem 10
2.5 Ordem dos cabeçalhos de extensão 12
2.6 Valores do campo *type* para mensagem ICMPv6 e respectivos tipos de mensagem 14
2.7 Relação de campos IPv4 - IPv6 sem fragmentação 19
2.8 Relação de campos IPv4 - IPv6 com cabeçalho de fragmentação 20
2.9 Relação de campos do cabeçalho de fragmentação IPv4 para IPv6 20
2.10 Relação de campos IPv6 - IPv4 sem fragmentação 21
2.11 Relação de campos IPv4 - IPv6 com cabeçalho de fragmentação 21
2.12 Diferenças entre as implementações stateless e stateful do NAT64 24
2.13 Relação de ferramentas utilizadas 29

3.1 Especificação de endereços das interfaces para o cenário da figura 3.10 40
3.2 Especificação de endereços das interfaces com endereçamento puramente IPv6 tendo como base a topologia da figura 3.10 41
3.3 Especificação de endereços das interfaces com endereçamento IPv4 tendo como base a topologia da figura 3.10 41
3.4 Especificação de endereços das interfaces com endereçamento IPv6 tendo como base a topologia da figura 3.10 42
3.5 Endereços IP de origem e destino utilizados nos testes de desempenho tendo como base a topologia da figura 3.10 44

4.1 Média de latência para as topologias IPv4-Only, IPv6-Only e NAT64 52
4.2 Valores de média de throughput para as topologias IPv4-Only, IPv6-Only e NAT64 53
4.3 Valores de média na variação de atraso para as topologias IPv4-Only, IPv6-Only e NAT64 em aplicação de áudio e vídeo de baixa qualidade 55
4.4 Valores de média na variação de atraso para as topologias IPv4-Only, IPv6-Only e NAT64 em aplicação de áudio e vídeo de média qualidade 56
4.5 Valores de média na variação de atraso para as topologias IPv4-Only, IPv6-Only e NAT64 em aplicação de áudio e vídeo de alta qualidade 57

LISTA DE SÍMBOLOS E SIGLAS

Siglas

QoS	*Quality of Service*
DNS	*Domain Name System*
NAT	*Network Address Translation*
DNS64	*Domain Name System 6to4*
NAT64	*Network Address Translation 6to4*
RFC	*Request For Comment*
ISO	*International Organization for Standardization*
IP	*Internet Protocol*
IPv4	*Internet Protocol version 4*
IPv6	*Internet Protocol version 6*
ICMP	*Internet Control Message Protocol*
ICMPv4	*Internet Control Message Protocol version 4*
ICMPv6	*Internet Control Message Protocol version 6*
UDP	*User Datagram Protocol*
TCP	*Transmission Control Protocol*
TTL	*Time To Live*
IHL	*Internet Header Length*
MF	*More Fragments*
FH	*Fragment Header*
NH	*Next Header*
ToS	*Type of Service*
CIDR	*Classless Inter-Domain Routing*
MSB	*Most Significant Bit*
IETF	*Internet Engineering Task Force*
BIB	*Binding Information Base*
RR	*Resource Records*
MSS	*Maximum Segment Size*
MTU	*Maximum Transfer Unit*
SNAT	*Source Network Address Translation*
DHCP	*Dynamic Host Configuration Protocol*
DHCPv6	*Dynamic Host Configuration Protocol version 6*
DNSSEC	*Domain Name System Security Extensions*
OSPF	*Open Shortest Path First*
BIND	*Berkeley Internet Name Domain*
VM	*Virtual Machine*

1 INTRODUÇÃO

Desde seu princípio, a rede mundial de computadores - a Internet - tem como principal objetivo proporcionar a todos os usuários serviços interoperáveis e de caráter universal, ou seja, proporcionar o acesso e a troca de informações por todos os sistemas e usuários integrantes da rede. Dessa forma, o desenvolvimento de novas tecnologias deve ter sempre como objetivo a manutenção de sistemas legados, buscando métodos que possibilitem a transição "amigável" das tecnologias previamente implementadas.

Com o esgotamento de endereços IP versão 4 em meados dos anos 90, fez-se necessária a implementação de técnicas que permitam a coexistência das novas redes IPv6 com a arquitetura IPv4 existente. O impacto causado na mudança do protocolo de rede e sua total implementação é um dos desafios encontrados atualmente em relação a internet, sendo então necessário um estudo aprofundado quanto a eficiência e eficácia deste processo. Como exemplo, este trabalho tem como pilar a análise de tráfego multimídia em redes que apresentam ambos os protocolos.

1.1 MOTIVAÇÃO

Uma das formas utilizadas para avaliação de eficiência e eficácia de novas técnicas é a simulação de ambientes de rede virtuais que utilizam uma estrutura de rede já implementada, de forma a proporcionar um ambiente com caráter configurável pelo usuário. Ferramentas *open-source* apresentam-se como uma alternativa viável neste tipo de modelagem, dado que estas tem como grande vantagem sua característica *"livre"*, ou seja, de aberta configuração por parte do usuário. Dessa forma, a construção de uma ferramenta que possibilite a fácil criação de uma rede personalizável vem a ser útil em diversas análises posteriores, podendo ser realizada a partir de scripts em Linux, em meio acadêmico.

1.2 OBJETIVO

O objetivo geral deste projeto consiste na criação de *shell scripts* em Linux que permitam a configuração automática de um ambiente de rede virtualizado com diferentes protocolos de rede e na análise de tráfego de rede neste ambiente com foco em qualidade de serviço (QoS) multimídia, utilizando o mecanismo de tradução de endereços (NAT64 / DNS64) IPv4 e IPv6 como forma de coexistência.

1.2.1 Objetivos Específicos

- Programação de *shell scripts* Linux que desempenhem configuração automática de hosts IPv4 e IPv6, roteadores de borda que implementam mecanismo de tradução NAT64, roteadores de rede interna e servidores DNS com mecanismo DNS64.
- Programação de scripts que implementem a funcionalidade de roteamento dinâmico (OSPF) e estático nos roteadores da rede, de forma a facilitar a configuração e propagação de rotas entre estes.
- Criação de uma interface de interação usuário-máquina via *dialog* para atribuição manual por parte do usuário de comandos e endereços.
- Realizar um estudo de caso em um ambiente de rede híbrido composto por máquinas virtuais e acesso à Internet, realizar testes de desempenho para tráfego multimídia via software *iperf* em uma arquitetura 6to4 e 4to6 como forma de validação da eficácia e do funcionamento adequado dos scripts implementados.

1.3 TRABALHOS CORRELATADOS

A tabela 1.1 apresenta alguns trabalhos recentes acerca da utilização do DNS64/NAT64 em redes 4in6 e 6in4. Não foram encontrados trabalhos acadêmicos relacionados à ferramentas de configuração para redes virtuais utilizando softwares livres.

Tabela 1.1: Trabalhos Correlatados

Título	Ano
Application compatibility of the NAT64 IPv6 transition technology [1]	2015
Packet Level TCP Performance of NAT44, NAT64 and IPv6 Using Iperf in the Context of IPv6 Migration [2]	2015
Provide IPv4 Service Using Pure IPv6 Servers with Stateless NAT64 Translator [3]	2014
Performance analysis and comparison of the TAYGA and of the PF NAT64 implementations [4]	2013
Performance Analysis and Comparison of Different DNS64 Implementations for Linux, OpenBSD and FreeBSD [5]	2013
IPv4/IPv6 transition using DNS64/NAT64: Deployment issues [6]	2012

1.4 ESTRUTURA DO TRABALHO

Este trabalho consiste em uma divisão baseada nas etapas necessárias ao entendimento, análise e conclusão acerca dos objetivos propostos. O capítulo de fundamentação teórica tem

como tópicos principais os protocolos IPv4 e IPv6, a técnica de NAT, servidores DNS e os mecanismos de coexistência para redes IPv4/IPv6, tendo como principal tópico o mecanismo de NAT64/DNS64. O capítulo 3 detalha a metodologia experimental aplicada a partir da proposta de uma ferramenta de configuração para redes virtualizadas e implementação de diferentes topologias para análise de tráfego em aspectos como tipo de endereçamento, objetivo, técnicas utilizadas, etc., além dos comandos aplicados nas máquinas virtuais. O capítulo 4 ilustra todos os resultados obtidos, assim como uma análise detalhada da eficácia e eficiência da implementação realizada. O capítulo 5 apresenta a conclusão obtida a partir dos resultados obtidos e propostas de projetos futuros a serem desenvolvidos tendo como base este trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 PROTOCOLOS E REDES IP

Em redes de computadores, para que seja garantida a comunicação entre os diversos dispositivos que as compõem, é necessária a utilização de uma arquitetura padrão que permita a interoperabilidade entre os diversos sistemas que utilizem os recursos disponíveis. A internet e sua arquitetura, conhecida como TCP/IP, baseia-se no modelo arquitetural de sete camadas proposto pela *International Organization for Standardization* (ISO). A arquitetura TCP/IP apresenta cinco camadas (aplicação, transporte, rede, enlace e física), onde cada uma apresenta funções específicas e fornece serviços para as camadas superiores [7].

O IP é um protocolo presente na camada de rede (camada 3) da arquitetura TCP/IP, responsável pelo endereçamento e roteamento dos pacotes que trafegam entre os roteadores e demais componentes da rede. A primeira versão deste protocolo é conhecida como IPv4. Posteriormente, dado iminente o esgotamento dos endereços IPv4, definiu-se o IPv6, versão que viria a substituir a anterior [7].

Por definição, uma rede IP é uma rede de computadores que trocam dados e informações em forma de pacotes utilizando como protocolo de comunicação o protocolo IP (Internet Protocol). Os sistemas finais representam dispositivos que geram o tráfego a ser transportado na rede (computadores, servidores, etc.), e no núcleo da rede, os datagramas IP trafegam por meio de roteadores que são responsáveis pelo roteamento destes até chegarem ao componente de rede de destino. A figura 2.1 ilustra de forma geral uma rede IP, composta pelos sistemas finais e núcleo, onde cada um desses sistemas fazem a utilização de camadas específicas da arquitetura TCP/IP, como indicado.

Para entendimento da proposta deste projeto, faz-se necessário um embasamento teórico em ambas as versões do protocolo IP (IPv4 e IPv6), que será apresentado nas subseções 2.1.1 e 2.1.3.

2.1.1 Protocolo IPv4

O protocolo IPv4 é especificado pela RFC 791, de setembro de 1981. O IP em sua concepção implementa duas funcionalidades dentro da rede: endereçamento, que consiste na atribuição de identificadores de 32 bits a cada um dos dispositivos de rede, e a fragmentação em datagramas menores que sejam suportados nos enlaces da rede [8]. Dessa forma, para um entendimento geral do funcionamento do IP, é necessária a divisão teórica de ambas as funções que o protocolo implementa.

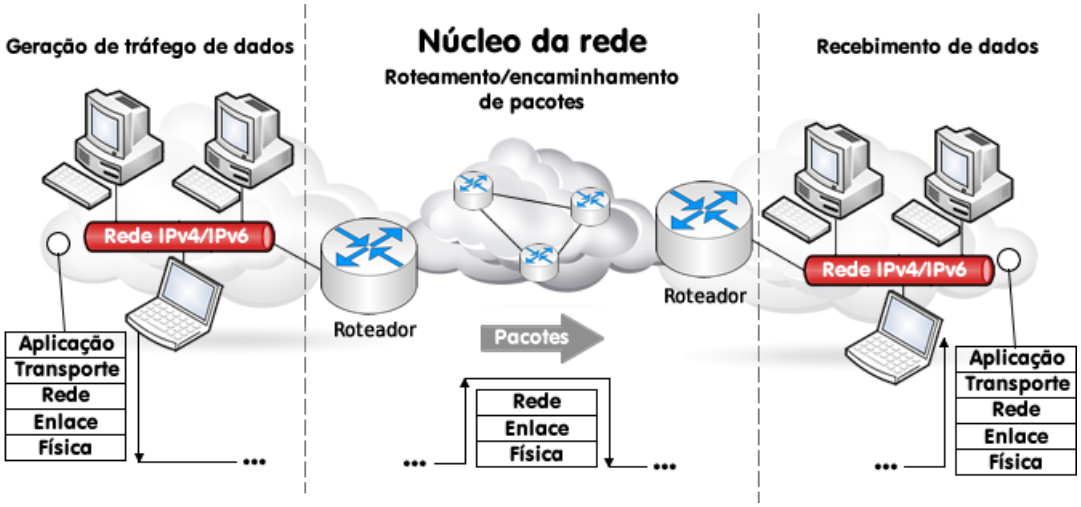

Figura 2.1: Ilustração geral de uma rede IP e das camadas TCP/IP
[Autores: Paulo Henrique Leal / Lucas Ribeiro]

2.1.1.1 Características gerais - Datagrama IP

Um datagrama IP é composto por duas partes: um cabeçalho (header) e a carga útil (payload). O cabeçalho é responsável pela identificação do endereçamento de origem e destino e estabelecimento de campos de controle. O payload contém o conteúdo a ser trafegado pela rede advindo das camadas de aplicação e transporte. Um cabeçalho IP é formado por um tamanho variado de bytes, sendo a quantidade mínima de 20 bytes, e a quantidade máxima, de 60 bytes. A figura 2.2 ilustra o formato e os campos do datagrama IP versão 4.

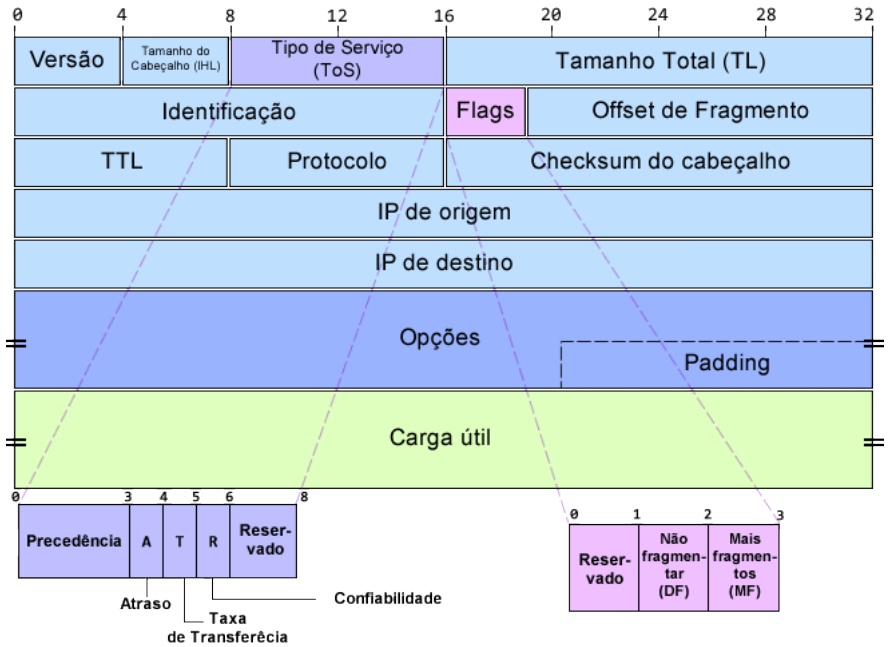

Figura 2.2: Formato do datagrama IP
[Autores: Paulo Henrique Leal / Lucas Ribeiro - adaptado de [8]]

Os campos do cabeçalho do datagrama IP apresentado na figura 2.2 apresentam as seguintes funcionalidades [9][10]:

- **Versão**: Versão do protocolo a ser utilizada, necessária para fins de compatibilidade e identificação do datagrama em dispositivos que trabalhem com ambas as versões. Apresenta 4 bits de tamanho, e valor 4 (0100) para a versão 4 e 6 (0110) para a versão 6 do IP.
- **Tamanho do cabeçalho (IHL)**: Campo de 4 bits com o propósito de informar a quantidade de bytes presentes no cabeçalho do datagrama. Esta variação na quantidade de bytes deve-se ao fato do campo *opções* ser opcional e apresentar tamanho variado. Caso o campo *opções* não seja utilizado, o IHL apresenta valor 5 (5×32 bits = 20 bytes), e por ser um campo de 4 bits, valor máximo 15 (15×32 bits = 60 bytes).
- **Tipo de Serviço (ToS)**: Campo de 8 bits utilizado para estabelecimento de parâmetros de qualidade de serviço, como prioridade, confiabilidade, atraso e taxa de transferência.
- **Tamanho Total**: Campo de 16 bits com o objetivo de informar o tamanho total **em bytes** do datagrama IP (cabeçalho + carga útil). Apresenta valor máximo de $2^{16} = 65536$ bytes.
- **Identificação / Flags / Offset de Fragmento**: Esses campos são responsáveis por fornecer informações referentes a fragmentação de datagramas. Uma vez que um datagrama pode vir a ser fragmentado antes de ser enviado e também chegar fora de ordem no destino, é necessária uma identificação e o estabelecimento de parâmetros que permitam a reconstrução e o ordenamento do datagrama original, sendo estes campos responsáveis por tal tarefa.
- **TTL (Time to Live)**: Campo de 8 bits responsável por indicar a quantidade de saltos restantes que o datagrama pode realizar na rede. O valor do campo é decrementado a cada salto na rede, e ao atingir valor nulo (TTL = 0), o pacote é descartado. Um dos principais objetivos é evitar *loops infinitos* na rede. Apresenta valor máximo de $2^8 = 256$.
- **Protocolo**: Campo de 8 bits responsável pela indicação do protocolo da camada superior a ser utilizado no trasporte dos dados. Os valores do campo são especificados na RFC 1700. Como exemplo, o TCP é representado pelo valor $(00000110)_2$ e o UDP pelo valor $(00010001)_2$, ambos protocolos da camada de transporte.
- **Checksum do cabeçalho**: Campo de 16 bits responsável por identificar a integridade do cabeçalho originalmente enviado, ignorando a carga útil. A cada salto, os roteadores realizam o cálculo da soma de verificação, que consiste em dividir o cabeçalho em

palavras de 2 bytes e somá-las, e caso essa seja diferente do valor presente no campo para o atual datagrama, esse é descartado.

- **IP de origem / IP de destino**: Campos de 32 bits que identificam os endereços IP de origem e destino do datagrama. Cada roteador na rede verifica o endereço de destino a fim de estabelecer uma saída (salto) que permita que este alcance tal endereço, e pode vir a modificar o endereço de origem também para tal fim.
- **Opções**: Campo de tamanho variado, com o objetivo de fornecer serviços e opções não definidas nos demais campos do cabeçalho do datagrama, oferecendo flexibilidade ao protocolo. É um campo **opcional**, podendo ter tamanho nulo (0 bytes).
- **Padding**: Componente do campo *opções* que possui como objetivo "preencher" este de forma a atingir-se uma quantidade de bits múltipla de 32 caso esta condição não seja satisfeita.

Definidos então os campos do cabeçalho, a este é anexada a carga útil a ser trafegada na rede, de forma que o datagrama apresente-se como "roteável" e alcance o componente de rede de destino seguindo a política de *melhor esforço* do protocolo IP.

2.1.1.2 Endereçamento

O protocolo IP utiliza como forma de endereçamento dos datagramas um número de 32 bits dividido em quatro octetos. Como observado no cabeçalho apresentado na figura 2.2, cada datagrama apresenta endereço de origem e destino. O endereço de origem é utilizado para fins de identificação da fonte de dados por parte dos componentes da rede (roteadores, *firewalls*, etc.) e do destinatário, que usualmente envia datagramas de resposta a este endereço. O endereço de destino é utilizado pelos roteadores para o estabelecimento de rotas na rede até que o datagrama chegue ao destino final.

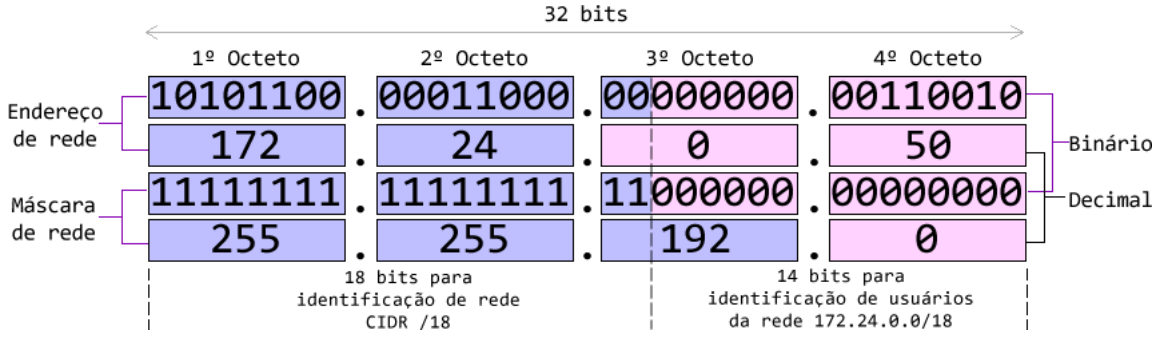

Figura 2.3: Estrutura e exemplo de endereço IPv4
[Autores: Paulo Henrique Leal / Lucas Ribeiro]

A figura 2.3 ilustra a estrutura de um endereço IPv4. Como observa-se na figura, há também a representação de uma máscara de rede no mesmo formato. Esse tipo de representação permite a divisão dos bits de endereçamento em bits de rede e de usuário, como

especificado na RFC 791. De forma hierárquica, os bits mais significativos são utilizados para identificação de rede, representados pelo bit 1 na máscara. Os bits menos significativos são utilizados para identificar os hosts/usuários, representados pelo bit 0 na máscara [8]. A notação **CIDR** é o padrão utilizado para representar um endereço IP e a rede a qual este pertence, ou simplesmente um bloco de endereços. O formato adotado é <Endereço IP>/<Máscara CIDR>, onde a máscara CIDR equivale a quantidade de bits de rede do endereço.

A RFC 791 especifica que endereços IPv4 são divididos em classes, de forma a possibilitar a flexibilidade em relação a divisão entre o número de redes e o número de usuários. A tabela 2.1 destaca a diferença entre as classes de endereços[8][11].

Tabela 2.1: Divisão de classes dos endereços IPv4

Classe	MSB	Faixa de endereços	Formato
A	0	0.0.0.0 - 127.255.255.255	7 bits de rede, 24 bits de hosts
B	10	128.0.0.0 - 191.255.255.255	14 bits de rede, 16 bits de hosts
C	110	192.0.0.0 - 223.255.255.255	21 bits de rede, 8 bits de hosts

Como observa-se na tabela 2.1, a divisão em classes possibilita a priorização da quantidade de hosts (classe A), de redes (classe C) ou um balanceamento entre ambos (classe B). São definidas também classes além das três citadas, reservadas para fins de pesquisa e serviços de *multicast*.

A toda rede IP são designados endereços de rede e de broadcast. O endereço de rede é representado sempre pelo primeiro endereço do bloco que representa a rede, e o de broadcast pelo último. Por exemplo, para o endereço 172.24.5.10/24, tem-se que a rede a qual este pertence é representada pelo endereço 172.24.5.0/24 com endereço de broadcast 172.24.5.255/24, onde o valor 0 no quarto octeto equivale ao primeiro endereço de uma rede com máscara 255.255.255.0 e 255 ao último.

Outra importante divisão existente entre endereços de redes diz respeito à endereçamento de redes privadas e públicas. Um endereço privado é considerado como "não roteável" na internet, uma vez que diversas redes privadas utilizam a mesma faixa de endereços em seus dispositivos. Sendo assim, uma rede privada é tida como uma rede composta por endereços pré estabelecidos que necessitam passar por um processo de tradução (NAT) antes de serem repassados para a rede externa. Endereços públicos são aqueles que identificam unicamente um host na rede, não podendo ser replicado ou reutilizado. A tabela 2.2[12] especifica a alocação estabelecida pela RFC 1918 para os endereços privados.

Os endereços especificados na RFC 1918 para alocação de redes privadas implementam a mesma política de divisão por classes, onde o bloco 1 consiste em apenas uma rede classe A, o bloco 2 consiste de 16 redes classe B, e o bloco 3 de 256 redes classe C. Posteriormente, neste documento, serão apresentados os conceitos do mecanismo de tradução de endereços (NAT) e a utilização de endereços privados será apresentada com maior clareza.

Tabela 2.2: Divisão de endereços privados

Bloco	Classe	Faixa de endereços	Prefixo CIDR
1	A	10.0.0.0 - 10.255.255.255	/8
2	B	172.16.0.0 - 172.31.255.255	/12
3	C	192.168.0.0 - 192.168.255.255	/16

2.1.2 Protocolo ICMPv4

O protocolo IP não foi desenvolvido como um protocolo confiável, logo, está suscetível a erros e demais problemas. O propósito do protocolo ICMP é trabalhar junto ao IP no envio de mensagens entre hosts, sendo essas mensagens relacionadas a informações de feedback no ambiente de comunicação [13].

O ICMP é transportado na rede por meio de um cabeçalho IP, onde os oito primeiros bits de informação ICMP definem o tipo de mensagem e consequentemente o restante da informação carregada. Os campos do cabeçalho IPv4 que carrega um pacote ICMP são definidos na tabela 2.3[13].

Tabela 2.3: Valores dos campos do cabeçalho IPv4 para mensagem ICMPv4

Campo	Valor
Versão	4
IHL	Tamanho do cabeçalho em palavras de 32 bytes
Tipo de Serviço	0
Tamanho total	IHL + Payload em bytes
Identificação, flags e offset	Variável
TTL	Número máximo de saltos
Protocolo	**1 (ICMP)**
Checksum	Variável
End. de origem e destino	Variáveis

O cabeçalho ICMP tem um formato pré definido e varia de acordo com cada mensagem. A figura 2.4 ilustra a estrutura de uma mensagem de erro ICMP, onde os bytes e cabeçalho do datagrama que enviou a mensagem são necessários para correta associação do processo.

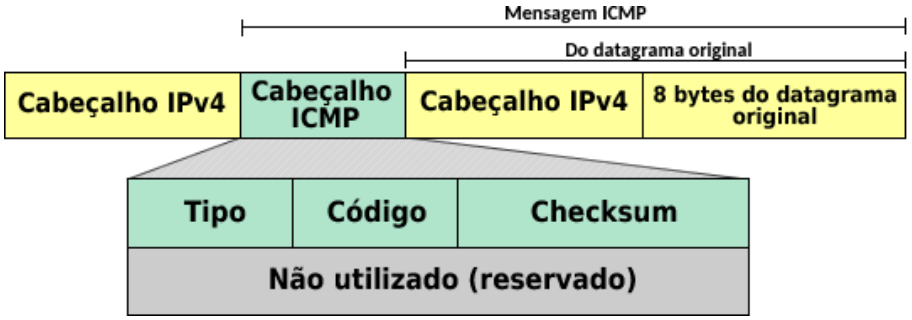

Figura 2.4: Estrutura de mensagem de erro ICMPv4
[Autores: Paulo Henrique Leal / Lucas Ribeiro]

A tabela 2.4 sumariza os tipos de mensagem ICMP que podem ser geradas [13].

Tabela 2.4: Valores do campo *type* para mensagem ICMPv4 e respectivos tipos de mensagem

Valor do campo *type*	Tipo de mensagem
0	Resposta Echo
3	Destino inalcançável
4	Redução de taxa de envio
5	Redirecionamento
8	Requisição Echo
11	Tempo excedido
12	Problema de parâmetro
13	Requisição de timestamp
14	Resposta de timestamp
15	Requisição de informação
16	Resposta de informação

2.1.3 Protocolo IPv6

A versão 6 do protocolo IP foi especificada no ano de 1998 pela RFC 2460, com a intenção de suceder o protocolo IPv4. Esta nova versão surgiu como uma alternativa ao esgotamento da quantidade de endereços IPv4 disponíveis e também com objetivos como simplificar o cabeçalho do datagrama IP, melhorar o suporte a extensões e opções e adicionar funcionalidades de melhora para sistemas em tempo real a partir da técnica de *Flow Labeling* [14].

2.1.3.1 Características gerais

O datagrama IPv6 apresenta significativas diferenças em relação a versão 4. A principal diferença está na simplificação do cabeçalho e no aperfeiçoamento do campo de opções. Desconsiderando os cabeçalhos de extensão, o cabeçalho IPv6 é formado por um tamanho fixo 40 bytes.

A figura 2.5 ilustra a estrutura de um datagrama IPv6. Como observa-se, a simplificação do cabeçalho baseia-se na redução da quantidade de campos existentes. Os campos do cabeçalho IPv6 apresentam as seguintes funcionalidades:

- **Versão**: Versão do protocolo a ser utilizada, idêntico ao campo presente no cabeçalho IPv4.
- **Classe de tráfego**: A classe de tráfego é dividida em duas partes, os 6 bits mais significantes são usados para estabelecer o ToS e os outros dois são usados para o mecanismo de *Explicit Congestion Notification (ECN)* [15].
- **Flow Label**: Campo de 20 bits que pode ser utilizado por uma fonte de tráfego para

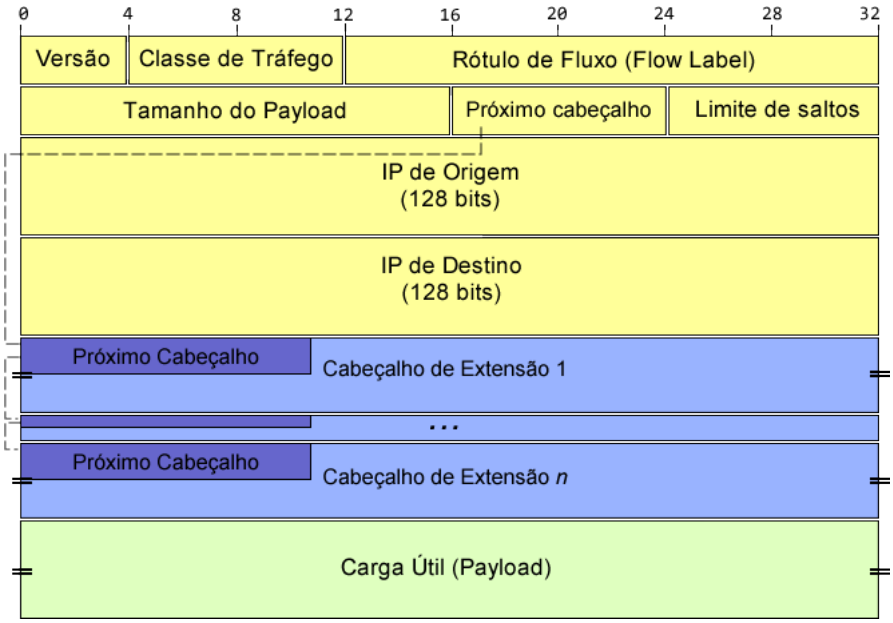

Figura 2.5: Estrutura do datagrama IPv6
[Autores: Paulo Henrique Leal / Lucas Ribeiro - Adaptado de [14]]

rotular sequências de pacotes que requerem tratamento especial pelos roteadores IPv6, como tratamentos específicos de QoS e serviços em tempo real [14].

- **Tamanho do Payload**: Utilizado para especificar a quantidade de octetos presentes no datagrama após o término do cabeçalho, ou seja, o tamanho em bytes da carga útil.
- **Próximo cabeçalho**: Utilizado para especificar o código do próximo cabeçalho de extensão ou a ausência destes.
- **Limite de saltos**: Indica a quantidade de saltos que o datagrama pode realizar na rede antes de ser descartado, funcionalidade idêntica ao TTL do IPv4.
- **IP de origem / IP de destino**: Campos de 128 bits que identificam os endereços IP de origem e destino do datagrama.

Para a manutenção das funcionalidades dos campos retirados em relação ao cabeçalho IPv4, o IPv6 utiliza de cabeçalhos de extensão que são anexados logo após o endereço de destino. Como observa-se na figura 2.5, pode-se adicionar antes da carga útil uma quantidade n de cabeçalhos de extensão, e os campos de "Próximo cabeçalho" apresentam o código referente ao cabeçalho seguinte. Entre as funcionalidades inseridas pelos cabeçalhos de extensão, estão opções diversas para os roteadores, funções de fragmentação, roteamento, mobilidade, criptografia e autenticação. Os cabeçalhos de extensão, de acordo com a RFC 2460, devem ser encandeados segundo a ordem apresentada na tabela 2.5 [16].

Ao cabeçalho IPv6 é adicionado então o *payload* advindo da camada superior e o datagrama é trafegado na rede de acordo com a política de melhor esforço do protocolo IP.

Tabela 2.5: Ordem dos cabeçalhos de extensão

Ordem	Tipo	Código N.H.
1	Cabeçalho IPv6 Básico	-
2	Opções Hop-by-Hop	0
3	Options de destino (com roteamento)	60
4	Cabeçalho de Roteamento	43
5	Cabeçalho de Fragmentação	44
6	Cabeçalho de Autenticação	51
7	Encapsulamento e segurança de payload	50
8	Opções de destino	60
9	Cabeçalho de mobilidade	135
-	Sem Next Header	59

2.1.3.2 Endereçamento

O protocolo IPv6 utiliza como forma de endereçamento um número de 128 bits, representado por 8 grupos de 16 bits, separados pelo símbolo ":" e representados em uma notação hexadecimal. Dessa forma, a quantidade de endereços possíveis em relação ao IPv4 aumenta em uma ordem de 2^{96}. Cada datagrama apresenta endereços de origem e destino, de forma a possibilitar o roteamento e a entrega desses ao correto destinatário.

Assim como nos endereços do protocolo IPv4, no IPv6 os bits mais significativos são utilizados para representar a rede a qual pertence o endereço, e os demais identificam individualmente cada host. A figura 2.6 ilustra um exemplo de endereçamento IPv6.

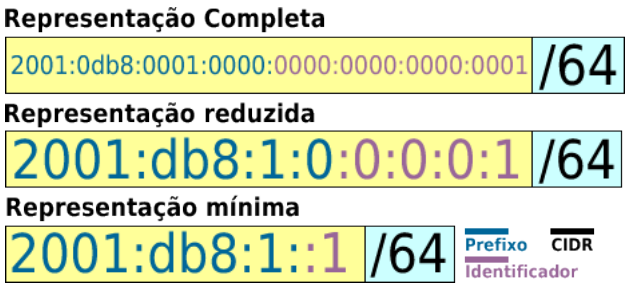

Figura 2.6: Exemplo de endereço IPv6 e diferentes representações possíveis
[Autores: Paulo Henrique Leal / Lucas Ribeiro]

Como observa-se na figura 2.6, o IPv6 utiliza também a notação CIDR para representar seus endereços, sendo assim, estes são identificados pelo endereço em si seguido pela máscara CIDR, representada por um valor decimal que especifica a quantidade de bits contíguos à esquerda do endereço que compreendem o prefixo. Percebe-se também que a representação dos blocos em hexadecimal pode ser feita de diferentes maneiras, todas identificadas igualmente pelos dispositivos de rede IPv6. A representação completa exibe todos os 8 blocos de 16 bits contendo todos os 4 dígitos hexadecimais que os compõem. A representação reduzida apresenta também todos os 8 blocos, mas neste caso, os dígitos hexadecimais "0" mais significativos são suprimidos, restando apenas o dígito referente aos

menos significativos. Na representação mínima, grupos de blocos contendo apenas dígitos "0" são substituídos pelos símbolos "::", sendo que este modelo pode ser utilizado apenas uma vez em cada endereço, para evitar ambiguidades na representação [17].

Em relação aos tipos de endereço IPv6, estes são divididos em 3 grupos: *unicast, anycast e multicast*, e diferentemente do IPv4, não existe um endereço de *broadcast*, sendo esses tratados por um tipo específico de multicast. Os três tipos são detalhados a seguir [17]:

- **Unicast**: identifica uma única interface, de modo que um pacote enviado a um endereço deste tipo é entregue a uma única interface;
- **Anycast**: identifica um conjunto de interfaces. Um pacote encaminhado a um endereço deste tipo é entregue a interface pertencente a este conjunto mais próxima da origem (de acordo com distância medida pelos protocolos de roteamento). Um endereço anycast é utilizado em comunicações de um-para-um-de-muitos.
- **Multicast**: também identifica um conjunto de interfaces, entretanto, um pacote enviado a um endereço deste tipo é entregue a todas as interfaces associadas a esse endereço. Um endereço multicast é utilizado em comunicações de um-para-muitos.

2.1.4 Protocolo ICMPv6

O protocolo ICMPv6 possui os mesmos propósitos do ICMPv4, especificados na subseção 2.1.2, porém, é implementado em redes que utilizam o protocolo IPv6 para comunicação. Sendo assim, as mensagens ICMPv6 são divididas em mensagens de erro e de informação [18].

- Mensagens de erro - mensagens relacionadas a erros de processamento, entre elas: Destino inalcançável, Tempo excedido, Pacote muito grande e Problemas de parâmetro.
- Mensagens de informação - mensagens relacionadas ao fornecimento de informações para gateways da rede, entre elas: mensagens de diagnóstico, *Neighbor Discovery* e mensagens de gerência de grupos multicast.

O formato do cabeçalho ICMPv6 assemelha-se ao ICMPv4, ilustrado na figura 2.4. A diferença está no fato do ICMPv6 ser incluído como um cabeçalho de extensão do cabeçalho IPv6, e não como corpo da mensagem. Dessa forma, o campo *Next Header / Próximo cabeçalho* de um datagrama IPv6 que contém uma mensagem ICMPv6 é setado para o valor 58, referente ao cabeçalho de extensão ICMPv6. A figura 2.7 ilustra a estrutura de uma mensagem IPv6/ICMPv6.

A tabela 2.6 define os possíveis valores do campo *type* do cabeçalho ICMPv6 e seus respectivos significados [18].

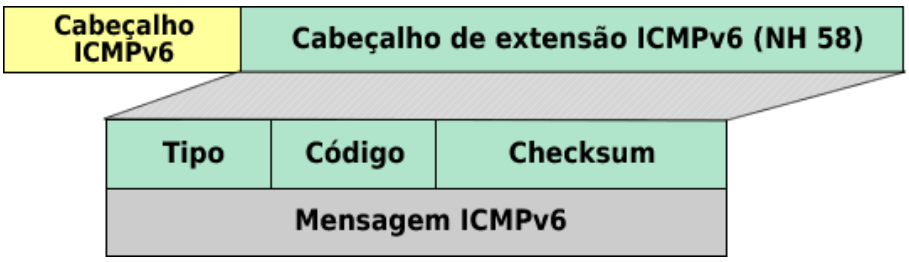

Figura 2.7: Estrutura de mensagem IPv6/ICMPv6
[Autores: Paulo Henrique Leal / Lucas Ribeiro]

Tabela 2.6: Valores do campo *type* para mensagem ICMPv6 e respectivos tipos de mensagem

Valor do campo *type*	Tipo de mensagem
1	Destino inalcançável
2	Pacote muito grande
3	Tempo excedido
4	Problema de parâmetro
128	Requisição Echo
129	Resposta Echo
130	Pergunta de membro de grupo
131	Resposta de membro de grupo
132	Redução de membro de grupo
133	Solicitação de roteador
134	Aviso de roteador
135	Solicitação de vizinho
136	Aviso de vizinho
137	Redirecionamento
138	Renumeração de roteador

2.1.5 Mecanismos de transição de redes IPv4 para IPv6

O esgotamento dos endereços IPv4 públicos ocorreu em 3 de fevereiro de 2011 [19], e em 2016 tem-se 77% dos hosts habilitados com o IPv6 [20]. De forma a proporcionar-se a coexistência e gradual migração do endereçamento IPv4 e IPv6, o IETF estabeleceu formas de permitir a interoperabilidade dos protocolos até a total habilitação dos dispositivos com o novo protocolo. Os principais meios são a pilha dupla de protocolos (*dual stack*), o tunelamento e a tradução. No tunelamento encapsula-se pacotes IPv6 dentro de pacotes IPv4 de forma a permitir a passagem de pacotes IPv6 em redes que funcionam unicamente com protocolo IPv4. A implementação da pilha dupla de protocolos significa operar com ambas as versões do IP, de forma que não apenas os *hosts*, mas também os equipamentos de rede devem possuir pilha dupla. A tradução consiste na conversão de endereços IPv6 em um pool de endereços IPv4. Todos esses mecanismos foram criados de forma a permitir a introdução de dispositivos IPv6 na rede de forma gradual até que os endereços IPv4 se esgotassem, mas isso aconteceu antes do esperado [19].

Os mecanismos de tunelamento e pilha dupla serão apresentados sumariamente nas sub-

seções 2.1.5.1 e 2.1.5.2, e o mecanismo de tradução utilizando NAT64/DNS64 será detalhado na seção 2.2, por apresentar-se como tema principal do projeto.

2.1.5.1 Tunelamento 4in6 e 6in4

O tunelamento IPv6 habilita hosts e roteadores IPv6 a se conectarem com demais componentes de rede IPv6 através de uma rede puramente IPv4 já existente e vice-versa. Esta técnica consiste em encapsular datagramas IPv6 em datagramas IPv4, fazendo com que os pacotes trafeguem pela rede IPv4 (como a internet) até que atinjam o correto destino. Para que isso funcione, os roteadores e dispositivos IPv6 devem estar pré configurados para realizar a o encapsulamento e desencapsulamento dos datagramas IPv6 de forma a encaminhá-los corretamente [21].

Através da utilização de túneis é possível o encapsulamento tanto de datagramas IPv4 em IPv6, quanto o contrário, e esses túneis são conhecidos como 4in6 e 6in4, respectivamente, e são ilustrados nas figuras 2.8 e 2.9.

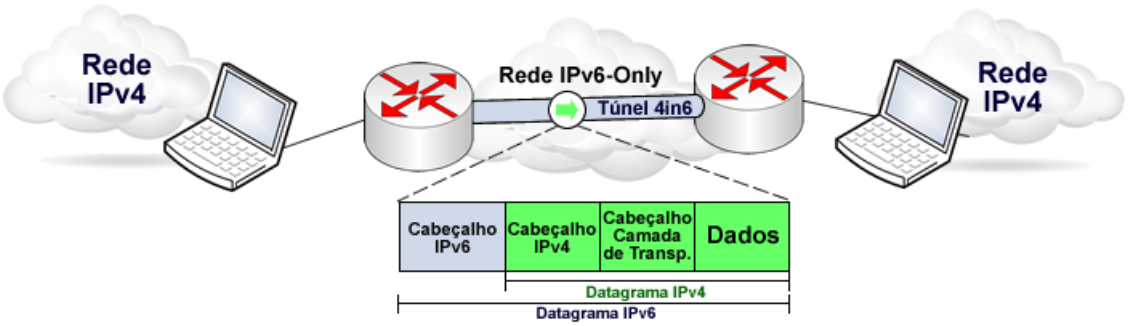

Figura 2.8: Exemplo de arquitetura de rede com tunelamento 4in6
[Autores: Paulo Henrique Leal / Lucas Ribeiro]

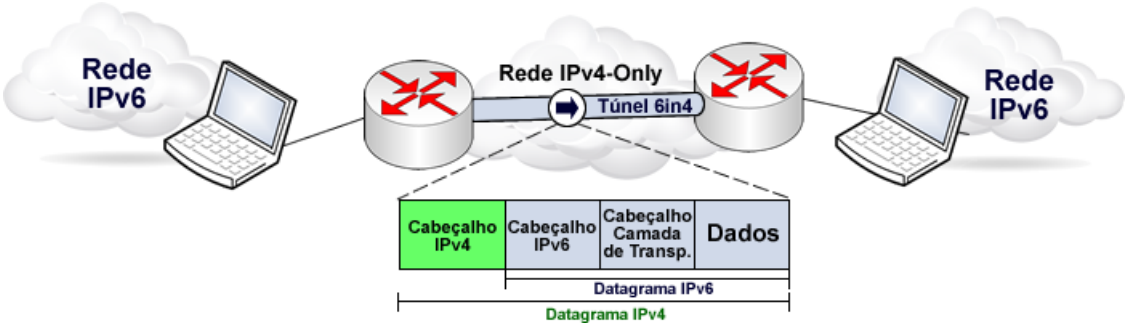

Figura 2.9: Exemplo de arquitetura de rede com tunelamento 6in4
[Autores: Paulo Henrique Leal / Lucas Ribeiro]

Como observa-se nas figuras 2.8 e 2.9, os pacotes que trafegam pela rede, seja esta puramente IPv4 ou IPv6, possuem seus dados mantidos (advindos da camada de transporte) e a estes são adicionados novos cabeçalhos IPv4 ou IPv6 de forma a possibilitar seu encaminhamento no núcleo da rede [22].

2.1.5.2 Pilha Dupla

O mecanismo de pilha dupla consiste em fornecer completo suporte aos protocolos IPv4 e IPv6 em todos os hosts e roteadores da rede. Dessa forma, garante-se a comunicação entre quaisquer nós IPv6/IPv4 (que implementam ambos os protocolos) com o restante da rede [22].

Por questões de configuração, pode-se optar por desligar o suporte a um dos protocolos. Sendo assim, os roteadores e hosts que implementam a pilha dupla podem operar em três modos: pilha IPv6 habilitada e pilha IPv4 desabilitada, pilha IPv4 habilitada e pilha IPv6 desabilitada, e ambas as pilhas habilitadas. Consequentemente, ao desabilitar uma das pilhas, o host ou roteador passa a ser tratado na rede como IPv4-Only ou IPv6-Only, igualmente aos demais que não implementam a pilha dupla [22].

Em relação ao endereçamento, hosts e roteadores que implementam a pilha dupla possuem ambos os tipos de endereço, IPv4 e IPv6, e utilizam estes para se comunicar com o restante da rede de acordo com o protocolo implementado pela rede de destino [22].

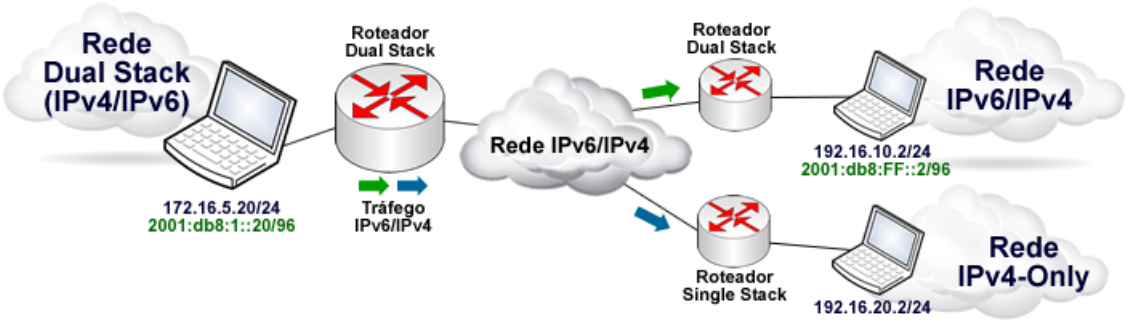

Figura 2.10: Exemplo de arquitetura de rede com implementação de pilha dupla
[Autores: Paulo Henrique Leal / Lucas Ribeiro]

A figura 2.10 ilustra um exemplo de arquitetura de rede que implementa a pilha dupla. No exemplo, um host pertencente a uma rede *dual stack* possui endereços IPv4 e IPv6 e é capaz de enviar através do núcleo da rede (também *dual stack*) pacotes IPv6 e IPv4 a outras redes IPv4-Only ou *dual stack*. Dessa forma, garante-se a comunicação e coexistência de dispositivos que implementam qualquer um dos protocolos.

2.2 O MECANISMO DE TRADUÇÃO IPV4/IPV6

2.2.1 NAT - Network Address Translation

A técnica de mapeamento de endereços conhecida como *Network Address Translation* (NAT) foi inicialmente especificada pela RFC 1631, mas deixava alguns aspectos de comportamento abertos, possibilitando que diferentes empresas que desenvolviam equipamentos de rede buscassem sua implementação específica. Dessa forma, causaram-se conflitos de domí-

nio e o mecanismo chegou a ser considerado prejudicial para o bom funcionamento da rede mundial [23]. Além de problemas de comunicação e desempenho, a RFC 1631 não definia o comportamento da tradução IPv4 para IPv6. De modo a uniformizar a tecnologia, foram definidos um conjunto de pré requisitos que devem ser cumpridos para que o NAT IPv4 possua correta interação com os protocolos ICMP, TCP e UDP. A RFC 1631 passou a ser considerada obsoleta e foi substituída pela RFC 3022.

De forma geral, o NAT IPv4 opera conectando dois domínios que têm endereços IPv4 diferentes, e comumente é utilizado para conectar um domínio com endereços privados a um de endereços públicos. Ao se receber um pacote com endereço de origem de um domínio privado (vide tabela 2.2), o NAT faz um mapeamento entre o par de **porta e endereço de origem** com o par de **endereço e porta pertencente ao seu *pool***, substituindo o primeiro pelo segundo e encaminhando o pacote à rede pública. O esquema supracitado é uma das formas que NAT tem de realizar o mapeamento. A figura 2.11 pode ser utilizada como base para análise dos tipos de mapeamento, onde A:a, B:b e C:c representam a tupla (endereço:porta) de cada host. No geral, são especificados três tipos de mapeamento [24]:

- ***Endpoint-Independent Mapping***: O mapeamento é feito com base no par endereço/porta dos pacotes dos hosts internos, ou seja, se todos os pacotes têm o mesmo endereço/porta, estes são sempre traduzidos para o mesmo endereço/porta da *pool* do NAT, independente do endereço/porta público de destino. Em relação a figura 2.11, os pares endereço/porta A1:a1 e A2:a2 seriam os mesmos para quaisquer pares B:b e C:c.

- ***Address-Dependent Mapping***: Nesse tipo de mapeamento analisa-se o endereço/-porta do domínio privado junto a apenas o endereço proveniente do domínio público para realizar o *binding*. Assim, pacotes que possuem o mesmo endereço/porta privados com destino ao mesmo endereço de domínio público recebem sempre o mesmo par endereço/porta pertencentes ao *pool* NAT. Em relação a figura 2.11, os pares endereço/porta A1:a1 e A2:a2 serão os mesmos se, e somente se, B for igual a C.

- ***Address and Port-Dependent mapping***: Mapeamento dependente de endereço e porta. Neste caso, analisa-se os pares endereço/porta privados e públicos, e se ambos os pares forem iguais, será atribuído sempre o mesmo endereço/porta do *pool* NAT. Em relação a figura 2.11, os pares endereço/porta A1:a1 e A2:a2 serão os mesmos se, e somente se, B:b for igual a C:c.

Apesar de realizar um mapeamento, o NAT não necessariamente deve encaminhar todos os pacotes provenientes de endereços privados e públicos. Com o NAT é possível a aplicação de regras de filtragem, de forma a aumentar a segurança da rede. Assim como no mapeamento, as regras de filtragem são definidas de acordo com o par endereço/porta. Os tipos de filtragem especificados são [24]:

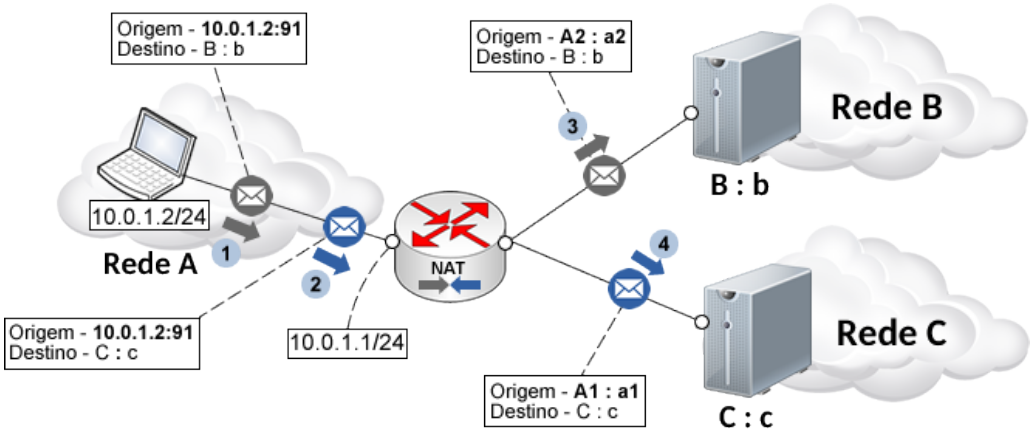

Figura 2.11: Esquema de mapeamento endereço/porta
[Autores: Paulo Henrique Leal / Lucas Ribeiro]

- ***Endpoint-Independent filtering***: A filtragem é feita baseada apenas no par endereço/porta dos hosts internos da rede NAT. Todos os pacotes não destinados a rede interna privada são filtrados.
- ***Address-Dependent filtering***: A regra de filtragem é aplicada ao par endereço/-porta dos hosts internos ao NAT, mas neste caso, também dependem do endereço IP do host externo. Essa regra estabelece que pacotes com destino a um host rede interna privada só são encaminhados caso este tenha antes enviado pacotes ao endereço requisitante.
- ***Address and Port-Dependent filtering***: A regra de filtragem depende do par endereço/porta dos hosts internos ao NAT e também do par endereço/porta externo. O comportamento é similar ao modo Address-Dependent filtering, mas neste caso é também necessário que o host da rede interna tenha enviado pacotes para o endereço e porta requisitante.

É recomendado ao NAT implementar o *endpoint independent mapping*, e em caso de filtragem, implementar o *endpoint independent filtering*.

2.2.2 NAT64 - Network Address Translation 6to4

O NAT64 é uma extensão do NAT. De forma geral, o NAT64 traduz o par endereço/-porta IPv6 para IPv4 e vice-versa. Existem dois mecanismos de tradução: a tradução de endereços, que consiste em associar um endereço IPv6 a um IPv4 e vice-versa, e a tradução de protocolo. A tradução de protocolos traduz os cabeçalhos IPv6 para IPv4 e vice-versa de forma a manter a semântica dos dados o mais fiel possível ao original.

2.2.2.1 Tradução de Cabeçalho/Protocolos

Em relação a tradução de cabeçalho IP, a RFC 6145 define um algoritmo a ser utilizado. Basicamente, ao realizar uma tradução deste tipo, o cabeçalho IPv6 é substituído por um IPv4 equivalente ou vice-versa e a parte de dados advinda das demais camadas permanecem inalteradas.

Uma das principais questões a ser analisada ao realizar-se uma tradução é a fragmentação de pacotes, com a intenção de não exceder-se os limites de MTU da rede. O IPv4, como ilustrado na figura 2.2, utiliza o campo de *flags* para definir a necessidade de fragmentação, e assim é possível a realização desta tarefa pelos roteadores na rede. Roteadores IPv6 não fornecem suporte a fragmentação de pacotes, apenas a fonte de dados pode realizar esta tarefa. Dessa forma, quando o bit DF (*Don't Fragment*) do IPv4 é setado, é realizada previamente uma descoberta de MTU através de mensagens ICMPv6 que também são traduzidas para serem corretamente interpretadas pelos roteadores IPv4, e assim garante-se que nenhum dos protocolos venha a exceder erroneamente o MTU das redes de destino. Caso o bit de DF não seja setado pelo protocolo IPv4, são utilizados então cabeçalhos de extensão de fragmentação nos cabeçalhos IPv6 que fragmentam os pacotes IPv4 com o valor mínimo de MTU em uma rede IPv6. É importante observar que o valor mínimo de MTU para o IPv4 é de 68 bytes, contra 1280 do IPv6, logo, pacotes IPv4 traduzidos para IPv6 não podem ser fragmentados com valor inferior ao MTU mínimo da rede IPv6 (e.g. 100 bytes, que satisfaz o requisito IPv4, mas não o IPv6) [25]. A figura 2.12 ilustra a mudança que ocorre nos datagramas ao realizar-se uma tradução IPv4 para IPv6 ou vice-versa.

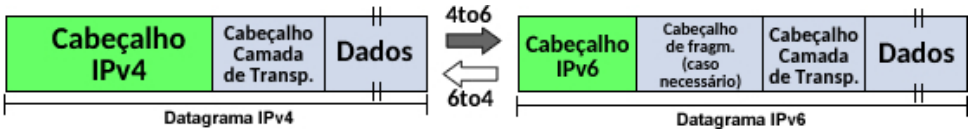

Figura 2.12: Processo de tradução 6to4 e 4to6
[Autores: Paulo Henrique Leal / Lucas Ribeiro]

No processo de tradução 4to6, os pacotes IPv4 traduzidos resultam em um tamanho superior a 1280 bytes. Isso significa que é necessária a fragmentação desses pacotes através da adição de cabeçalhos de fragmentação ao pacote IPv6 resultante. Os campos do cabeçalho IPv6 são então configurados de acordo com o que é apresentado na tabela 2.7 [26].

Tabela 2.7: Relação de campos IPv4 - IPv6 sem fragmentação

Campo IPv6	Valor traduzido do campo IPv4
Versão	Alterado de 4 para 6
Classe de Tráfego	Cópia do campo Type of Service*
Flow Label	Todos os bits em 0
Tamanho do Payload	TL - (IHL + OL)**
Próximo cabeçalho (NH)	ICMPv4 (1) para ICMPv6 (58) ou Protocolo
Limite de saltos	Cópia do TTL***

* Em casos onde a semântica do ToS difere da Classe de Tráfego, todos os bits são setados em 0.

** TL: Total Length ; IHL: Internet Header Length ; OL: Options Length

*** Como o tradutor é um roteador, esse valor deve ser verificado e/ou alterado segundo as regras do protocolo IP antes de prosseguir com o encaminhamento do pacote.

O processo de tradução dos campos de endereço de origem e destino serão detalhados na seção seguinte. A parte de opções, que pode ou não estar presente no cabeçalho IPv4, não é traduzida para o cabeçalho IPv6, sendo esta ignorada.

Em relação a necessidade de adição de um cabeçalho de Fragmentação após a tradução, caso se aplique, são feitas as associações exibidas nas tabelas 2.8 e 2.9 [25].

Tabela 2.8: Relação de campos IPv4 - IPv6 com cabeçalho de fragmentação

Campo IPv6	Valor traduzido do campo IPv4
Tamanho do Payload	TL + 8 - (IHL + OL)
Próximo cabeçalho (NH)	Cabeçalho de Fragmentação (44)

Tabela 2.9: Relação de campos do cabeçalho de fragmentação IPv4 para IPv6

Campo IPv6	Valor traduzido do campo IPv4
Próximo cabeçalho (NH)	ICMPv4 (1) para ICMPv6 (58)
Offset de fragmentação	Cópia do campo IPv4
Mais Fragmentos	Cópia do campo MF IPv4
Identificação	Cópia do campo de identificação IPv4

No processo de tradução 6to4, os pacotes IPv6 traduzidos precisam também lidar com a questão da fragmentação. Caso haja a presença de um cabeçalho de fragmentação no cabeçalho IPv6, o tradutor é responsável por limpar o bit *Don't Fragment* do cabeçalho IPv4 resultante e copiar o valor de identificação do cabeçalho de fragmentação para o identificador IPv4, além de especificar a presença ou não de mais fragmentos. Os campos do cabeçalho IPv4 são então configurados de acordo com o que é apresentado na tabela 2.10 [26].

Em relação à tabela 2.10:

* Em casos onde a semântica do ToS difere da Classe de Tráfego, todos os bits são setados a um valor especificado pelo tradutor.

** O tradutor deve gerar um identificador e setar o bit DF para zero caso o tamanho do pacote seja maior que 88 e menor ou igual a 1280 bytes, para evitar *black holes* por filtragens ICMP [25].

*** Como o tradutor é um roteador, esse valor deve ser verificado e/ou alterado segundo as regras do protocolo IP antes de prosseguir com o encaminhamento do pacote.

**** Em caso de fragmentação, a tradução é feita de acordo com o que foi especificado no parágrafo anterior.

Em relação a presença de um cabeçalho de Fragmentação no pacote IPv6, caso se aplique, são feitas as associações exibidas na tabela 2.11 [25].

Tabela 2.10: Relação de campos IPv6 - IPv4 sem fragmentação

Campo IPv4	Valor traduzido do campo IPv6
Versão	Alterado de 6 para 4
IHL	5 (sem opções)
Type Of Service	Cópia da Classe de Tráfego *
Tamanho total	Payload IPv6 + Cabeçalho IPv4
Identificação	0 ou variável **
Flags	DF = 1 ; MF = 0 **
Offset de fragmento	0
TTL	Cópia do Limite de Saltos ***
Protocolo	ICMPv6 (58) para ICMPv4 (1) ou NH ****
Checksum	Calculado após tradução

Tabela 2.11: Relação de campos IPv4 - IPv6 com cabeçalho de fragmentação

Campo IPv4	Valor traduzido do campo IPv6
Tamanho Total	Payload IPv6 - 8 + Cabeçalho IPv4
Identificação	Identificação do FH
Flags	MF (IPv4) = M (IPv6) ; DF = 0
Offset de fragmento	Offset de fragmento do FH
Protocolo	ICMPv6 (58) para ICMPv4 (1) ou último cabeçalho IPv6

2.2.2.2 Tradução de Endereços

A tradução de endereços opera de forma a traduzir o par endereço/porta IPv6 para IPv4 e vice-versa. O NAT64 possui duas *pools*, uma IPv4 para representar o par endereço/porta IPv6 dentro da rede puramente IPv4, e uma *pool* IPv6 para representar os endereços/portas IPv4 na rede puramente IPv6.

A tradução de endereços de um domínio IPv4 utilizando NAT64 é feita usando-se um prefixo IPv6 do tipo "Prefixo64::/n" pertencente ao *pool* IPv6, onde n é a mascara de sub rede. Cada endereço IPv4 é mapeado em um IPv6 diferente que segue este modelo. Se n for menor que 96, é realizado então um *zero padding* no sufixo do endereço de forma a alcançar-se os 128 bits do endereço IPv6[27]. A RFC 6052 sugere um prefixo "64:ff9b::/96" para o propósito específico da tradução de endereços IPv4 para IPv6, mas também permite configuração manual de um prefixo que pertença ao bloco de endereços IPv6 de *unicast global*. Ao utilizar o prefixo conhecido (64:ff9b::/96) para uma representação IPv4 em IPv6, o endereço traduzido é reconhecido globalmente, logo, a Internet reconhece os pacotes como uma tradução IPv4 para IPv6 e consegue alcançar o endereço IPv4 caso haja um roteador NAT64 presente na rede. A figura 2.13 ilustra diferentes endereços IPv6 mapeados a partir de diferentes prefixos e suas respectivas estruturas [27].

No NAT64, a *pool* IPv4 não necessariamente contém muitos endereços e o mapeamento de IPv6 para IPv4 é feito de um para um de forma dinâmica. Quando o NAT64 relaciona um endereço/porta de um *host* IPv6 com um endereço/porta da sua *pool* IPv4, cria-se

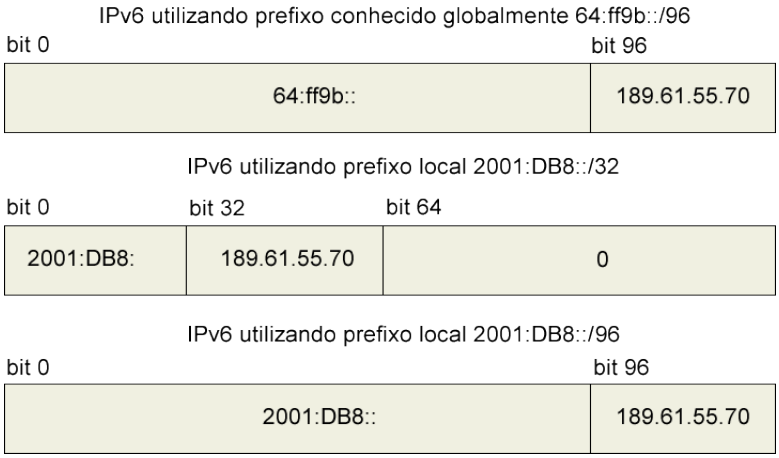

Figura 2.13: Representação IPv6 do endereço IPv4
[Autores: Paulo Henrique Leal / Lucas Ribeiro - Adaptado de [28]]

um *binding state*, de forma que todos os pacotes respondidos pelo *host* IPv4 podem ser traduzidos e encaminhados corretamente para o emissor IPv6. O *binding state* é mantido enquanto há fluxo de pacotes, e assim que este é finalizado, o *binding state* expira e o par endereço/porta IPv4 retorna a *pool* IPv4 do NAT64.

Além do *binding state*, o NAT64 utiliza a *Binding Information Base* (BIB), formada exclusivamente com informações de mapeamento da origem IPv6 para um destino IPv4. Quando inicia-se uma comunicação que não tem um registro de endereço/porta na tabela BIB, uma nova entrada é criada. A tabela BIB é utilizada na filtragem do tipo *Address Independent Filtering*, assim como a *Session Table* que armazena os endereços dos hosts IPv4. Essa tabela contém os endereços/portas de origem e destino IPv6, assim com os endereços/porta de origem e destino IPv4. Cada entrada tem um *life time*, e no caso dos protocolos UDP e TCP, estes valores são definidos nas RFCs [29] e [23]. As conexões UDP tem *life time* de 2 minutos e as TCP de 2 horas. Por conta do grande custo de *life time* das conexões TCP, o NAT64 monitora a conexão, e quando esta é finalizada, retira-se o *binding* de forma a liberar o endereço/porta para a *pool* IPv4 [28].

A figura 2.14 ilustra um exemplo de funcionamento do NAT64 para o estabelecimento de uma conexão IPv6 com a Internet IPv4. As seis etapas, são detalhadas a seguir.

1: Um host da rede puramente IPv6 envia uma requisição a um host IPv4 utilizando como endereço de destino um endereço no formato (PREFIXO/96)::(IPv4).

2: O roteador com NAT64 habilitado identifica o endereço de destino no formato (PREFIXO/96)::(IPv4) e encaminha-o ao túnel NAT64, que realiza um *binding* com um endereço do *pool* IPv4 e retira o prefixo do endereço de destino. Neste caso, o *pool* IPv4 do NAT64 pode ser 192.168.255.0/24.

3: Opcionalmente, para conexão com a internet, o roteador NAT64 pode realizar também um SOURCE NAT, associando um endereço/porta ao pacote, de forma a enviá-lo com um endereço reconhecido pela rede externa.

4: Ao receber a resposta à requisição antes realizada pelo host IPv6, o roteador realiza a verificação do endereço/porta de destino e verifica a tabela NAT para correta recuperação do endereço ao qual realizou o SOURCE NAT.

5: O roteador NAT64 verifica a tabela de *binding* e extrai o endereço IPv6 de destino do pacote recebido, além de acrescentar o prefixo NAT64 ao endereço de origem.

6: Os pacotes são recebidos pelo host IPv6 da rede interna.

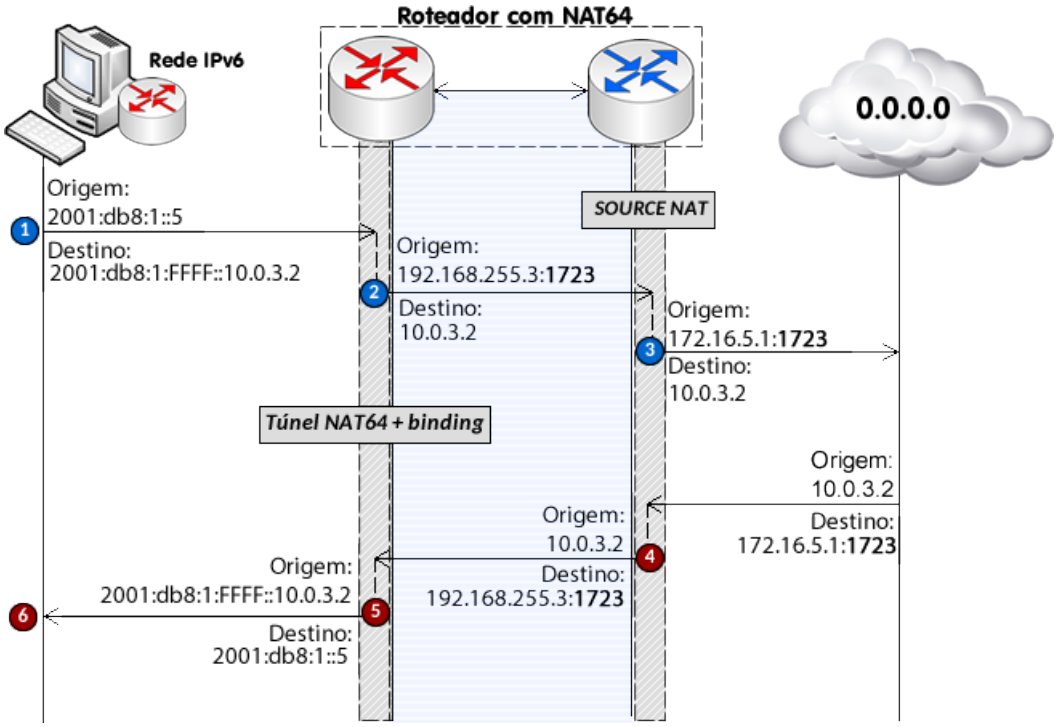

Figura 2.14: Exemplo de sequência de funcionamento do NAT64
[Autores: Paulo Henrique Leal / Lucas Ribeiro]

2.2.2.3 NAT64 Stateless e Stateful

Em relação à tradução de endereços e portas, o IETF específica duas formas de se realizar este processo, a tradução *stateless*, especificada na RFC2765, e a tradução *stateful*, que é que especifica na RFC6145.

A tradução *stateless* no NAT64, como o nome diz, não guarda estado da conexão, pois o mapeamento de porta e endereço é feito de 1:1. Sendo assim, se um cliente de um domínio com endereçamento puramente IPv6 quiser se comunicar com um domínio IPv4, um endereço IPv6 será mapeado para um endereço IPv4, e futuramente toda vez que esse cliente tentar qualquer comunicação com o domínio IPv4, este receberá o mesmo endereço. Esse método tem a vantagem de ser rápido, além de não requisitar grandes recursos da maquina hospedeira do NAT64, pois exige um mapeamento simples. Entretanto, não resolve o problema da exaustão de endereçamento IPv4, pois se a rede IPv6 for muito grande, rapidamente a *pool* de endereços IPv4 do NAT64 será esgotada. Por fim, é obrigatório o

uso do DHCPv6 ou a configuração manual dos endereços de interface IPv6.

Na tradução *stateful* executada pelo NAT64, cria-se um estado no servidor para cada tradução, permitindo uma tradução de 1:N. Essa técnica permite a multiplexação de vários dispositivos IPv6 em um único endereço IPv4 para se comunicar com a rede mundial. O processo funciona muito bem quando um dispositivo com endereço IPv6 inicia uma comunicação para a rede IPv4, mas se um dispositivo IPv4 quiser utilizar serviços de uma rede IPv6 e iniciar a comunicação, é necessário configurações adicionais no NAT64 [25]. A tabela 2.12[30] sumariza as principais diferenças entre o ambos os tipos de tradução.

Tabela 2.12: Diferenças entre as implementações stateless e stateful do NAT64

Stateless	**Stateful**
Tradução 1:1	Tradução 1:N
Não conserva endereços IPv4	Conserva endereços IPv4
Garante transparência e escalabilidade fim-a-fim	Utiliza overloading de endereços, logo, possui baixa transparência
Não mantém estado, nem realiza *bindings* na tradução	Mantém estado e faz *binding* nas traduções
Requer associação válida de endereços IPv6 com IPv4	Não há restrição de associação
Requer endereçamento manual ou via DHCPv6	Aceita qualquer tipo de endereçamento manual/dinâmico

2.2.3 DNS64 - Domain Name System 6to4

O *Domain Name System 64* (DNS64) funciona de forma a traduzir *resource records* (RR) do tipo AAAA de RRs do tipo A. Ou seja, o DNS64 permite que um host pertencente a uma rede com endereçamento puramente IPv6 use um nome de domínio totalmente qualificado (*Fully Qualified Domain Name*) de um *host* IPv4 para iniciar uma comunicação.

O algoritmo de funcionamento do DNS64 é baseado em realizar buscas de ambos os tipos, A e AAAA. Quando um DNS64 recebe uma consulta RR do tipo AAAA gerada por um host IPv6, ele procura por um RR AAAA para o endereço solicitado. Caso o endereço tipo AAAA seja recebido e a consulta DNS validada, o DNS64 envia diretamente ao host IPv6 o resultado e armazena em cache este endereço IPv6. Caso não haja RR (NXDOMAIN) do tipo AAAA para resolvê-lo (e.g. um host em uma rede puramente IPv4), o DNS64 faz uma consulta procurando um RR do tipo A. Se um RR do tipo A é descoberto, o DNS64 então cria um RR do tipo AAAA sintético adicionando o prefixo do NAT64 “Prefixo64::/n” ao endereço IPv4 descoberto para o host IPv6 que iniciou a consulta. Se “n” for menor que 96, é feito um zero *padding* no sufixo. Após a descoberta, o RR do tipo AAAA então é repassado para o *host* IPv6 que inicia uma comunicação IPv6 utilizando o endereço associado ao host IPv4 consultado [28]. A figura 2.15 ilustra um exemplo do processo descrito neste parágrafo.

De forma a aumentar a segurança das consultas de resolução de nomes, o DNS64 tem

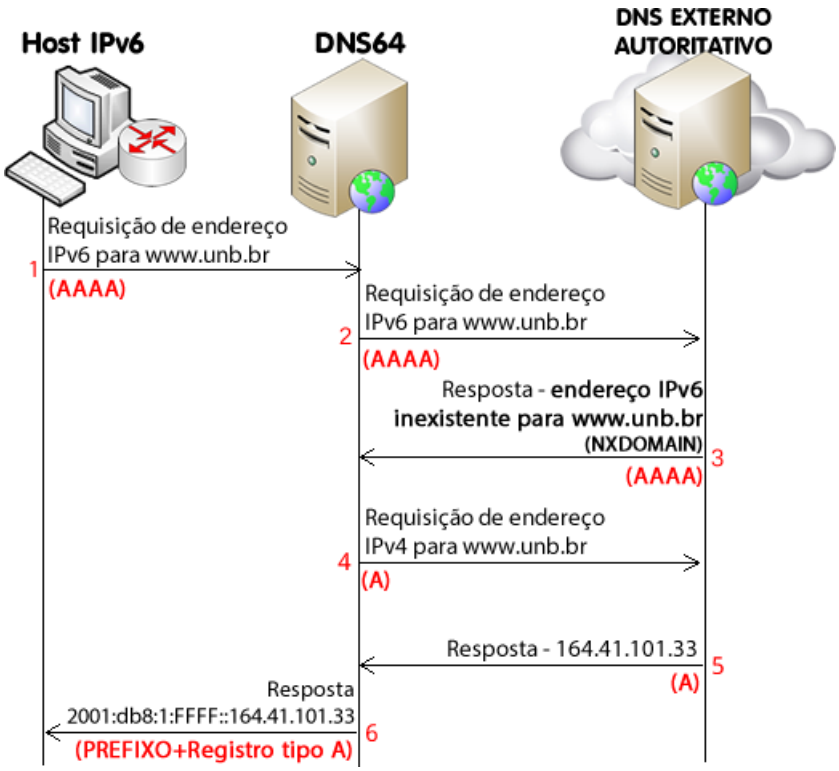

Figura 2.15: Exemplo de sequência de funcionamento do DNS64
[Autores: Paulo Henrique Leal / Lucas Ribeiro]

compatibilidade com o DNSSEC. O DNSSEC prova a autenticação da origem de um pacote, além de garantir a integridade dos dados DNS. Entretanto, o DNSSEC em todas as suas formas deve ser implementado no DNS64 de forma cuidadosa. Para evitar conflitos de autenticação, a síntese de RR deve ocorrer prioritariamente após a a validação do DNSSEC.

Existem diferentes configurações do DNSSEC para resolvedores recursivos no caso do DNS64. Um resolvedor recursivo pode ser capaz de efetuar o DNSSEC ou não, pode ser capaz de efetuar o DNSSEC e efetuar a validação dos dados, ou simplesmente pode repassar os dados do DNSSEC. Logo, o resolvedor recursivo DNS64 lida com as seguintes consultas DNS [28][31]:

1) Consulta de uma origem sem DNSSEC habilitado e sem validação: Neste caso, o *host* com endereçamento puramente IPv6 não está habilitado com DNSSEC, logo, quando recebe o RR do tipo AAAA sintetizado pelo DNS64, não há validação.

2) Consulta de uma origem habilitada com DNSSEC, mas sem validação de fonte: Nesta situação, se o DNS64 estiver com a validação de DNSSEC implementada, após obter exito na consulta do RR, este valida os dados DNSSEC, cria o RR sintético do tipo AAAA, e encaminha a consulta para a origem. Nesse tipo de configuração, o *host* espera que o DNS valide os dados DNSSEC, e é a forma recomendada de se implementar DNSSEC em um ambiente NAT64/DNS64, pois cria um canal seguro entre o cliente e o servidor DNS protegido por **IPsec**.

3) Consulta de uma origem habilitada com DNSSEC, mas sem validação de fonte: Com essa configuração, a origem da requisição de consulta pergunta pelos dados DNSSEC de forma que ela mesma realiza a validação. Por conta disso, o DNS64 não pode repassar um RR sintético do tipo AAAA antes da validação, pois assim esta falharia. Nesse caso, é recomendado colocar a função de DNS64 no próprio *host* de origem para que assim seja possível realizar a devida validação.

A figura 2.16 ilustra o funcionamento dos três tipos de consulta e validação DNS64 com DNSSEC.

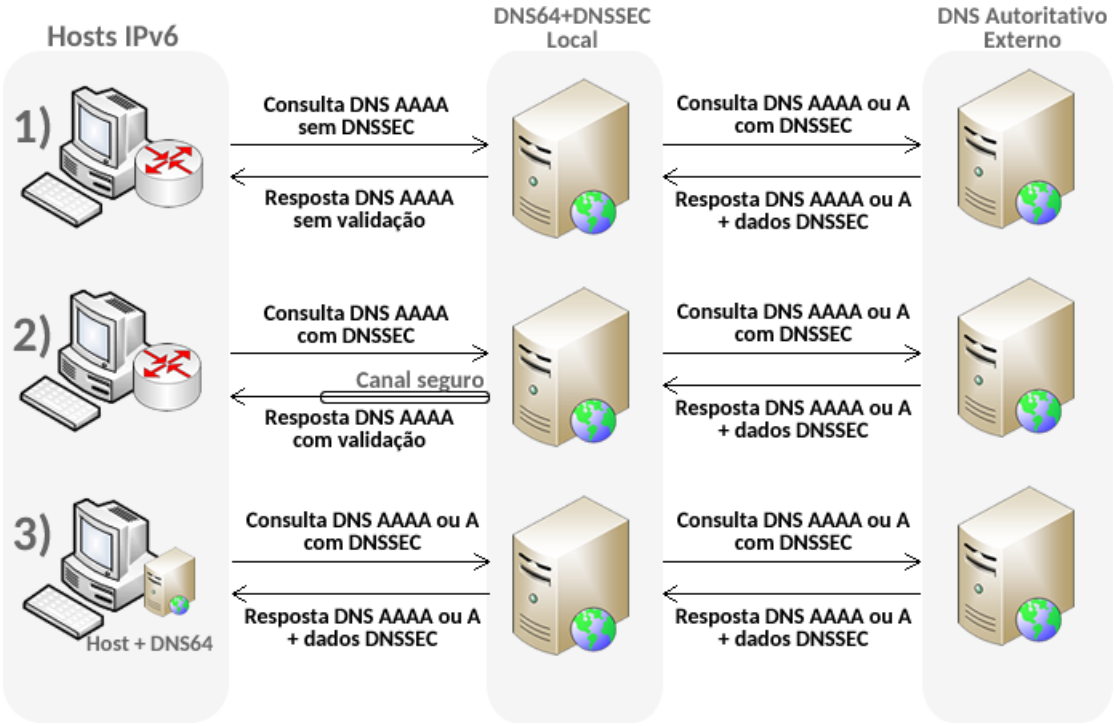

Figura 2.16: Tipos de validação DNS64 com DNSSEC
[Autores: Paulo Henrique Leal / Lucas Ribeiro - Adaptado de [28]]

O DNS64 opera juntamente ao NAT64 em um ambiente de rede IPv6 que utiliza o mecanismo de tradução como forma de coexistência com a rede IPv4 existente. Primeiramente, um host da rede realiza uma consulta ao servidor DNS64, que retorna um endereço IPv6 que pode referir-se a um endereço originalmente IPv6 ou a um endereço do tipo PREFIXO::IPv4/96 resultante de uma conversão de endereço. Em seguida, o host envia pacotes ao roteador NAT64 com o endereço de destino recebido, que identifica o formato do endereço e retira o prefixo NAT64 caso necessário. A figura 2.17 ilustra um exemplo deste processo [31].

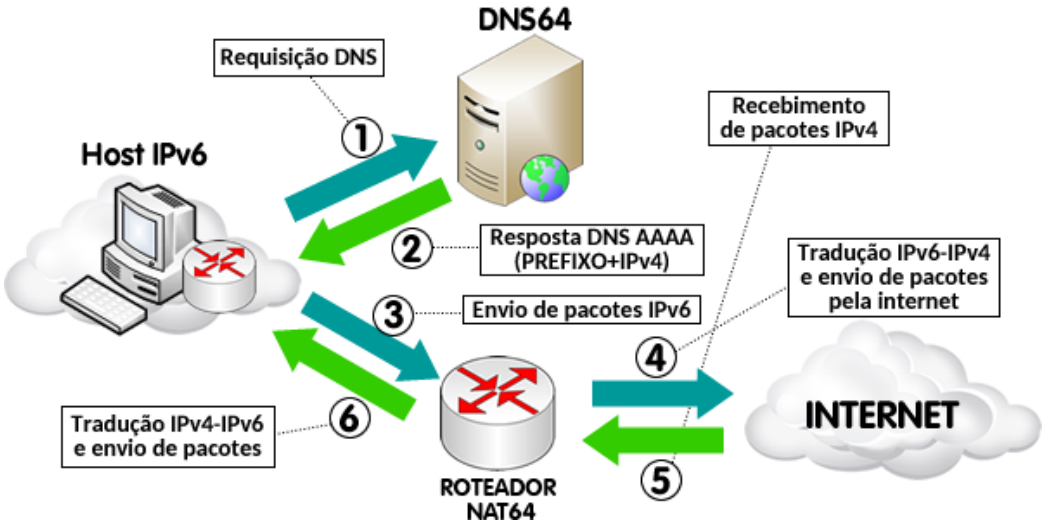

Figura 2.17: Exemplo de sequência de funcionamento do DNS64 junto ao NAT64 para endereço traduzido [Autores: Paulo Henrique Leal / Lucas Ribeiro]

2.3 AMBIENTES DE REDE VIRTUALIZADOS

A virtualização de máquinas apresenta-se como uma opção ao melhor aproveitamento de recursos de hardware em um ambiente computacional. Ao criar-se uma ou mais máquinas virtuais, conhecidas como *guests*, dentro de um sistema operacional, conhecido como *host*, ocorre o compartilhamento de recursos de hardware como memória, disco e processamento por parte de ambas as máquinas. Dessa forma, a virtualização pode ser vista como uma alternativa de menor custo e maior facilidade de administração de ambientes computacionais diversos, incluindo ambientes de rede [32]. O virtualizador *VirtualBox* possui uma versão Open Source, chamada VirtualBox OSE (Open Source Edition) sob licença GPL. Essa solução é capaz de emular discos rígidos a partir de um recipiente especial chamado Virtual Disk Image (VDI), mas também é capaz de emular outros formatos, como o popular ISO.

Em termos de virtualização de redes, a utilização de máquinas virtuais que operem como componentes de rede (roteadores, *switches*, *firewalls*, etc.) apresenta-se como uma alternativa viável para análise, experimentos e até mesmo fornecimento de serviços em ambientes de redes variados. Outro aspecto importante em relação a este tipo de implementação consiste na característica isolada que cada máquina virtual apresenta, o que implica que cada uma destas tem funcionamento e administração de recursos independentes, podendo ser utilizadas como componentes reais em um ambiente de rede, garantindo uma fidelidade superior a simuladores de redes convencionais.

A figura 2.18 ilustra resumidamente uma arquitetura de rede composta por máquinas virtuais. A máquina hospedeira compartilha recursos de memória, disco e processamento com as máquinas virtuais executadas dentro do ambiente do software de virtualização e comunica-se com as mesmas por meio de uma interface de rede virtual. Dessa forma, pacotes advindos da rede externa são repassados às máquinas virtuais de forma transpa-

rente, podendo essas então comunicarem-se com demais redes. O software de virtualização *VirtualBox* cria também uma rede interna utilizada para comunicação entre as máquinas virtuais que não estão conectadas diretamente ao hospedeiro por meio de uma interface virtual, permitindo a troca de mensagens entre todas os componentes da rede virtual [33].

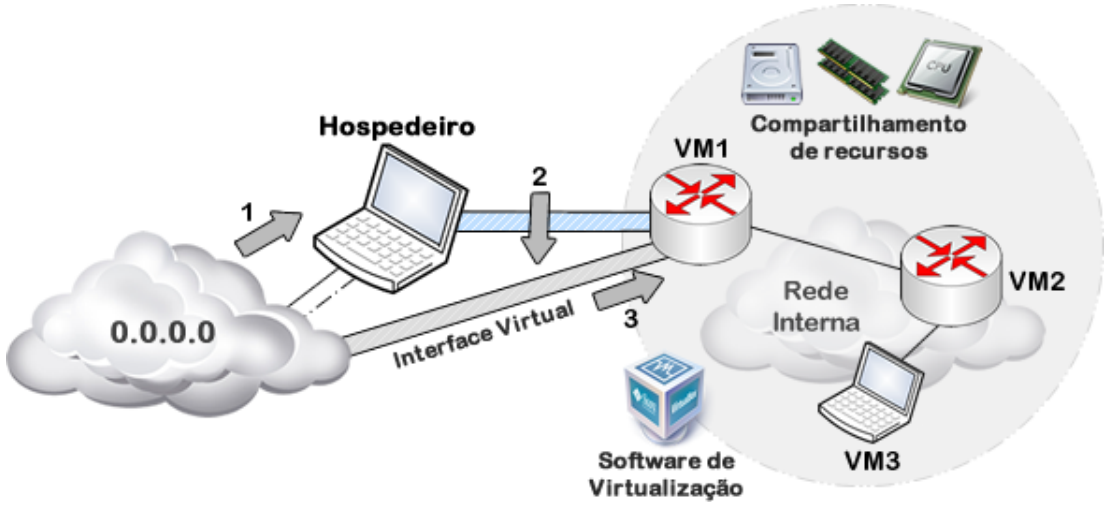

Figura 2.18: Arquitetura de rede virtual resumida
[Autores: Paulo Henrique Leal / Lucas Ribeiro]

2.4 AMBIENTES LINUX

Dada a característica de sistemas operacionais baseados no kernel Linux serem livres, ou seja, de *código aberto*, diversas implementações de softwares de rede estão disponíveis para testes, simulações e criação de ambientes variados. Em termos de processos de rede, sistemas operacionais Linux utilizam políticas de repasse de pacotes baseadas em processos / softwares que são executados em background no software, como *iptables* para filtragem, BIND para serviços DNS, etc.

Ao receber um pacote advindo de um determinado host ou componente de rede, o sistema operacional Linux verifica suas políticas de repasse, e caso possível, realiza o repasse / roteamento do pacote em questão. A figura 2.19 ilustra um processo de roteamento descrito em um sistema operacional Linux [34].

2.4.1 Ferramentas utilizadas

A tabela 2.13 especifica as ferramentas utilizadas para construção do script e demais verificações de desempenho. As principais ferramentas são detalhadas nas subseções 2.4.1.1 a 2.4.1.3.

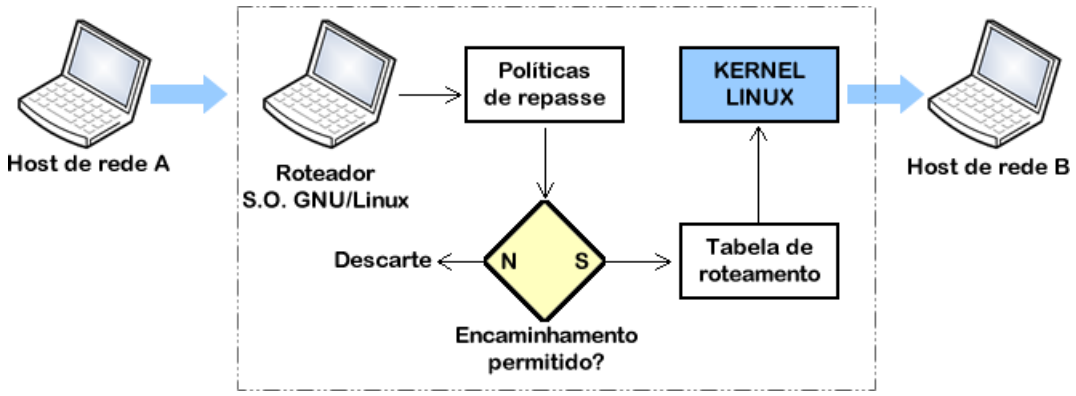

Figura 2.19: Processo de roteamento em ambiente GNU/Linux
[Autores: Paulo Henrique Leal / Lucas Ribeiro]

Tabela 2.13: Relação de ferramentas utilizadas

Ferramenta	Versão
Iperf	3.1.3
Tayga	0.9.2
Quagga	1.1.0
iptables	1.6.0
BIND	9.10.3
Dialog	1.3.2
Wireshark	2.2.0
VirtualBox	5.1.8

2.4.1.1 IPTables

O *iptables* é um programa de linha de comando compatível com distribuições Linux de Kernel 2.4 ou superior, e funciona de forma a prover serviços de *Firewall*, *NAT* e modificação de pacotes para sistemas linux.

O *iptables* funciona de forma a receber comados que definem regras, regras essas que podem ser definidas em sequência formando uma cadeia de comandos *(chains)* que aplicam filtragem nos pacotes tomando como base o IP e a porta de origem e destino. Outro elemento importante do *iptables* são as tabelas, que são onde salvam-se as *chains* e suas respectivas regras. Quanto um pacote entra no *host*, as tabelas são consultadas para verificar quando devem ser aplicadas as regras, para então depois analisar-se as opções de tratamento especificadas na linha de comando. Na versão mais atual do *iptables*, versão 1.6, existem ao total 5 tabelas, que são:

- ***Filter***: quando nenhuma especificação de tabela for passada na cadeia de comandos, as regras são aplicadas nessa tabela. Contém as chains *INPUT* destinadas para sockets locais, *FORWARD*, que consulta pacotes que são encaminhados para outras interfaces de rede, e *OUTPUT* para pacotes gerados localmente que estão saindo do host.
- ***nat***: tabela consultada quando um pacote que inicia uma nova conexão é encontrado.

Consiste das *chains PREROUTING*, usada para pacotes que precisam ser encaminhados assim que chegarem ao host, *OUTPUT*, consultada para pacotes gerados localmente antes do roteamento, e *POSTROUTING*, para pacotes que estão saindo do host, após as regras de roteamento.

- ***mangle***: tabela usada especialmente para alteração de pacotes, principalmente alteração de cabeçalho de pacote. Essa tabela funciona somente a partir da versão de *Kernel* Linux 2.4.18. Tem como *chains* a *PREROUTING*, usada para pacotes que precisam ser encaminhados assim que chegam ao host, *OUTPUT*, consultada para pacotes gerados localmente antes do roteamento, *POSTROUTING*, consultada para pacotes que estão saindo do host após as regras de roteamento, *INPUT*, para pacotes enviados para o iptables em si, e *FORWARD*, usada para pacotes que estão sendo roteados pelo host.

- ***raw***: utilizada para filtragem de conexões.

- ***security***: usada para aplicar controle utilizando endereços MAC, é consultada após a aplicação das políticas de filtragem especificadas na tabela *filter*. Possui as *chains INPUT*, destinadas para sockets locais, *FORWARD*, consultada para pacotes que são encaminhados para outras interfaces de rede, e textitOUTPUT, consultada para pacotes gerados localmente antes do roteamento.

Na linha de comando *iptables* também podem ser especificadas opções, que são divididas em dois grupos, os comandos e os parâmetros. Os comandos especificam qual ação deve ser feita, e só pode ser usada uma dessas opções em cada linha de comando. Os parâmetros fazem especificação de onde serão aplicadas as regras ou os comandos, como por exemplo, especificar qual a versão IP será usada.

2.4.1.2 NAT64 - Tayga

O *Tayga* é uma implementação do NAT64 para ambientes Linux, não sendo uma implementação de Kernel, devendo ser instalado via repositório ou binários. Para o funcionamento do NAT64, deve inserir-se nos arquivos de configurações do Tayga o prefixo IPv6, que será usado na tradução de endereços, assim como a *pool* de endereços que serão utilizados para representar um host IPv6 na rede IPv4. Tendo definido a *pool*, é preciso indicar um endereço IPv4 para o Tayga por questões de requisições e respostas ICMP, endereço esse que deve estar dentro da *pool* IPv4 definida anteriormente.

O Tayga opera a tradução NAT64 de forma *stateless* e traz as vantagens e desvantagens supracitadas na tabela 2.12, logo, no mapeamento de cada endereço IPv6 tem-se um único endereço IPv4 para sua representação na rede mundial IPv4. O mapeamento *stateless* traz flexibilidade, pois caso seja necessário um único endereço para representar toda uma rede IPv6 é possível utilizar NAT via *iptables*, utilizando a política de *MASQUERADE* ou

SNAT, dando assim uma maior flexibilidade na configuração e adição de redes e hosts em uma topologia IPv6 que fará uso do NAT64. No Tayga é permitido também a configuração manual de mapeamento de endereços IPv4 para um host IPv6, o que vem a ser útil para atribuição de endereços de servidores. O Tayga utiliza de um arquivo de formato *.map* para salvar as associações IPv6-IPv4 realizadas.

De forma a trocar pacotes com o Kernel do sistema, o Tayga utiliza do *driver TUN*. Em sua versão 0.9.2, o Tayga é compatível com as versões de Kernel 2.4, 2.6 ou superiores. Como a comunicação com o Kernel é feita utilizando um túnel, antes de levantar o serviço é preciso criar o mesmo, indicando os endereços das interfaces que se comunicarão com o *TUN*. Como configuração suplementar, deve se adicionar rotas de forma estática informando que, para se comunicar com a rede de prefixo e com a rede IPv4 definida pelo *pool* no Tayga, deve utilizar-se a interface criada no *TUN*.

2.4.1.3 DNS64 - BIND9

O *Berkley Internet Name Domain* (**BIND**) implementa o protocolo DNS, e é composto por:

- ***Domain Name Resolver***: O BIND possui seu próprio *resolver stub*, que encaminha as *queries* de nome de domínio para o *cache* do *resolver* e também para um servidor ou um grupo de servidores na rede dedicados ao serviço DNS. Desse forma, esse módulo tem por função repassar as *queries* para um ou múltiplos servidores autoritativos, de forma a encontrar o endereço IP para o nome de domínio requisitado.
- ***Domain Name Authority Server***: Além de ter um resolvedor *stub*, o BIND também possui seu próprio servidor autoritativo DNS que responde aos *queries* usando como base de informação os nomes de domínio.
- ***Ferramentas***: O BIND possui um conjunto de ferramentas de diagnóstico e operação, como a *DIG tool*, essa que não é especifica do BIND, mas pode ser utilizada para diagnosticar qualquer servidor DNS.

O BIND na sua versão 9 implementa o DNS64, onde o alvo é operado com consulta a origem sem o DNSSEC habilitado e sem validação DNSSEC.

Para que o BIND9 opere como um servidor DNS64, é preciso informar que ele deve retornar um RR do tipo AAAA se o cliente que fez a requisição pelo nome de domínio pertencer a uma *pool* específica e se o endereço IP daquele *hostname* não possuir RR do tipo AAAA. Esse RR AAAA é construído pelo BIND9 usando um prefixo, que obrigatoriamente é o mesmo prefixo utilizado no NAT64 presente na rede local. Quando um host tenta se conectar com um endereço IP com o prefixo informado ao BIND9, ele será encaminhado para o servidor NAT64 [35].

3 METODOLOGIA EXPERIMENTAL E ESTUDO DE CASO

3.1 PROPOSTA DE FERRAMENTA DE CONFIGURAÇÃO PARA REDES VIRTUALIZADAS

De forma a facilitar a configuração das maquinas virtuais que compõem a topologia de rede a ser utilizada, desenvolveu-se uma ferramenta utilizando a linguagem *bash*. Além de *bash*, foi utilizado o recurso do pacote *dialog*, como pode ser visto na figura 3.1, onde encontra-se o menu principal da ferramenta, que precisa ser instalado no ambiente *Linux* para o devido funcionamento do *script* implementado.

Com exceção do pacote *dialog*, o *script* é composto por comandos nativos *Linux*, que não precisam de instalação de pacotes adicionais e funcionam em ambiente de linha de comando. A ferramenta tem como funções principais a configuração das interfaces de rede em um host/roteador, a adição de rotas na tabela de roteamento, tanto IPv4 quanto IPv6 e realização de roteamento OSPF utilizando-se da ferramenta *Quagga*, que deve ser previamente instalada para uso correto da função. Como a ferramente realiza comandos e altera configurações da máquina *GNU/Linux*, ela deve ser executada em modo de super usuário.

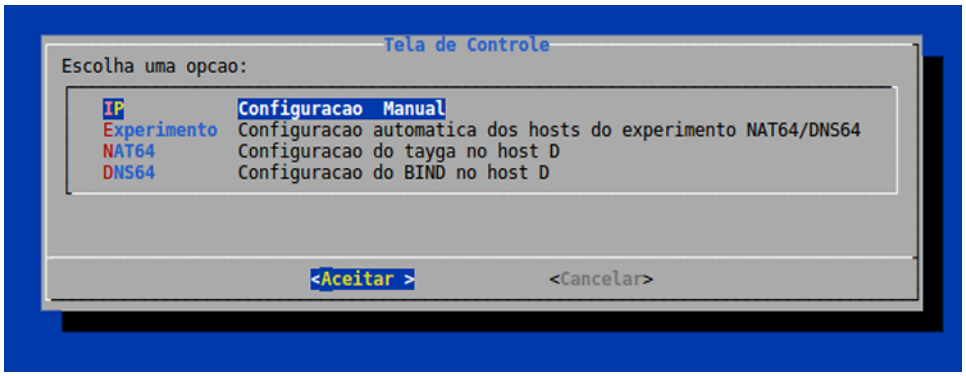

Figura 3.1: Menu principal da ferramenta de configuração

3.1.1 Configuração da interface de rede

Utilizando-se a ferramenta proposta, existem 2 meios de se configurar interfaces de rede. Uma das formas utiliza de comandos nativos do *Linux* para exercer tal função, e a outra utiliza o software *Quagga*, que faz uso do software *Zebra* contido no próprio pacote.

Ao escolher a opção IP no primeiro menu *dialog*, figura 3.1, abre-se a opção de configuração utilizando dos comandos nativos do sistema *Linux*, onde é possível escolher entre configuração de interface em endereçamento IPv4 e IPv6. Ao escolher umas das duas opções, escolhe-se a interface desejada para configuração e depois digita-se o endereço IP desejado com a máscara de rede CIDR. Vide figura 3.2.

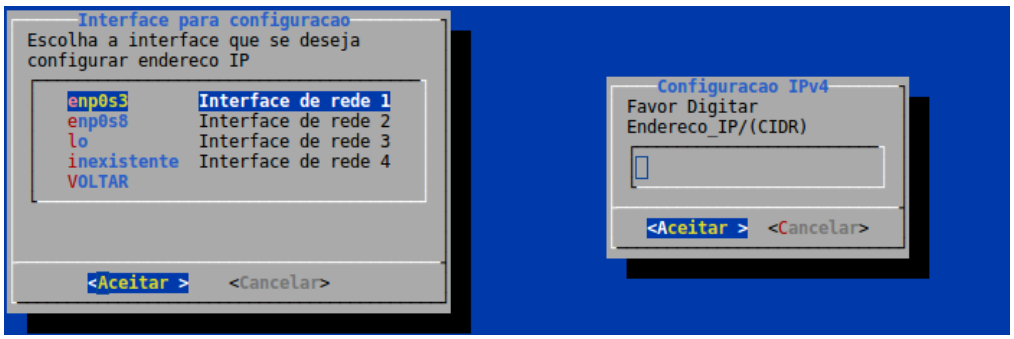

Figura 3.2: Menus da configuração de interface de rede

Em *background*, ao informar as opções pedidas pela ferramenta, são executados os comandos:

- Comando para limpar qualquer endereço previamente configurado na interface.

```
$ sudo ip addr flush dev <interface de rede>
```

- Comando para habilitar a interface, caso ela não esteja habilitada.

```
$ sudo ip link set <interface de rede> up
```

- Por fim, há a configuração propriamente dita da interface desejada com seu respectivo endereço IP.

```
$ sudo ip  addr add <endereço IP> dev <interface de rede>
```

- Para endereçamento IPv6:

```
$ sudo ip -6 addr add <endereço IP> dev <interface de rede>
```

3.1.2 Roteamento estático

No *dialog* da figura 3.3, é possível escolher a opção de configuração de roteamento, onde é possível adicionar rotas para redes com endereçamento IPv4 e IPv6. Escolhendo um dos tipos de roteamento, é possível configurar tanto rotas para uma rede específica quanto um rota *default*, como ilustra a figura 3.4. Para efetiva configuração de uma rota *default*, é preciso informar o endereço de rede do próximo salto, assim como a interface de saída para o endereço desejado.

Na configuração de uma rota *default*, a ferramenta executa para IPv4 o comando:

```
#ip route add default via <IP do próximo salto> dev <interface de rede >
```

E para IPv6:

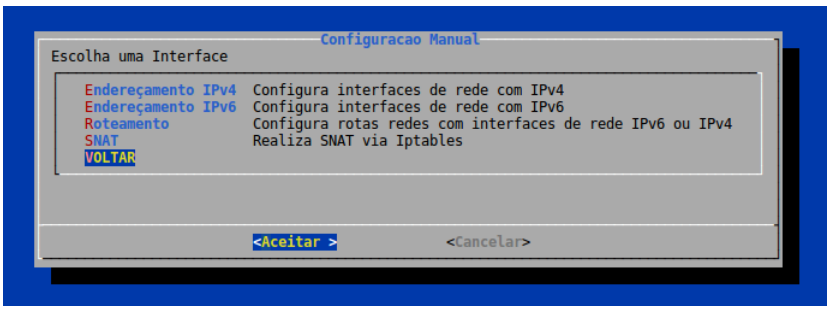

Figura 3.3: Menus da configuração de roteamento estático

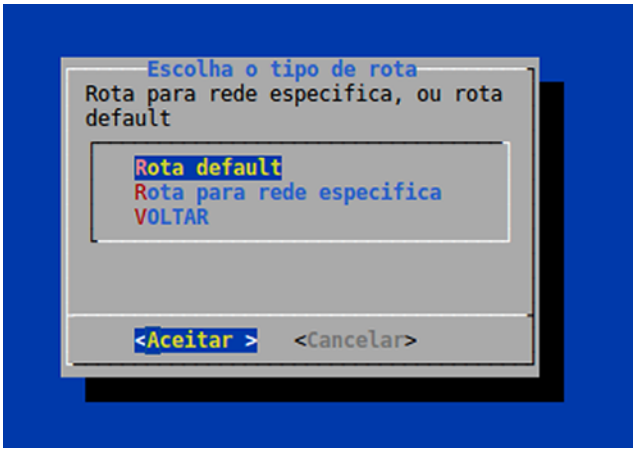

Figura 3.4: Menus da escolha do tipo de rota

```
#ip -6 route add default via <IP do próximo salto> dev <interface de rede >
```

No caso da adição de uma rota para uma rede desejada, deve-se informar o endereço de rede da rota desejada, endereço da interface de rede do próximo salto e a interface diretamente conectada a esse endereço.

Ao se adicionar uma rota para uma rede específica, é executado o comando:

```
#ip route add <endereço IP da rede de destino> via \
<endereço IP do próximo salto> dev <interface de rede >
```

Para redes com endereçamento IPv6:

```
#ip -6 route add <endereço IP da rede de destino> \
via <endereço IP do próximo salto> dev <interface de rede>
```

3.1.3 Configuração via Quagga

As configurações de roteamento e interface de rede podem ser efetuadas escolhendo a segundo opção da tela *dialog* inicial, figura 3.5.

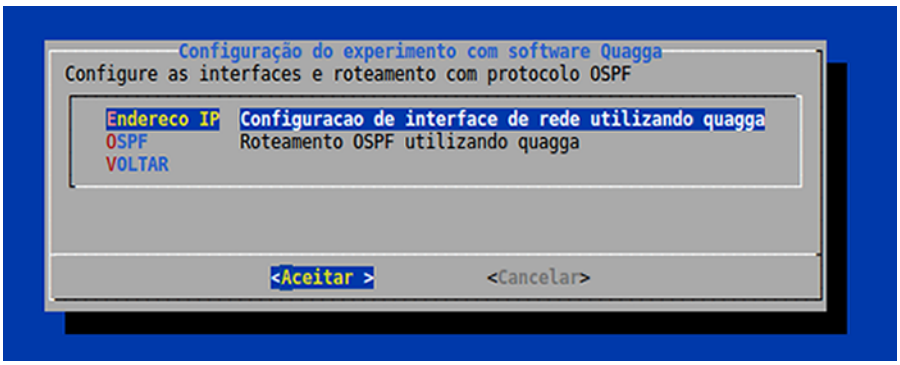

Figura 3.5: Menus da configuração do módulo Quagga

3.1.3.1 Configuração de interface via Quagga

No menu *dialog* para a configuração das interfaces de rede, é necessário informar quais interfaces devem ser configuradas e o endereço de rede de cada interface escolhida. Ao se digitar as informações, o par endereço de rede e interface são usados para alterar os arquivos de configuração *zebra.conf* do *Zebra*, que é o modulo do *quagga* responsável pela configuração das interfaces de rede, aquivo esse localizado no diretório:

```
/etc/quagga
```

3.1.3.2 Roteamento dinâmico com protocolo OSPF

Caso selecione-se a opção de configuração de roteamento OSPF, serão configurados no *quagga* os mesmos pré requisitos que roteadores de mercado exigem para a configuração do protocolo OSPF, ilustrados na figura 3.6. É preciso informar as redes diretamente conectadas, as interfaces que estão conectadas as redes informadas e a área que cada rede pertence. No caso da escolha do OSPF em endereçamento IPv6, também é preciso informar a identidade da maquina ou dispositivo que está sendo utilizado.

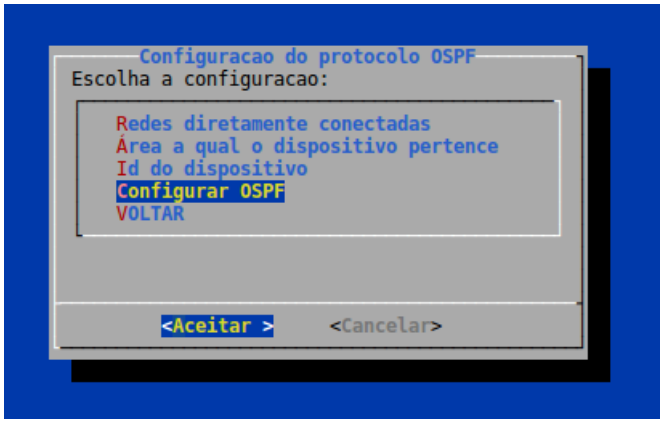

Figura 3.6: Menus da configuração do roteamento OSPF via Quagga

Tendo informado todos os parâmetros necessários para o OSPF com o protocolo IPv4, as

configurações desejadas são escritas no aquivo *ospfd.conf*. Para o OSPFv6, as configurações são escritas no arquivo *ospfd6.conf*.

Ambos os arquivos de configuração estão localizados na pasta de arquivos de configuração *quagga*:

```
/etc/quagga
```

3.1.4 NAT64

A ferramenta proposta efetua o NAT64 utilizando o software *Tayga*. Ao selecionar a opção de configuração NAT64, é aberto o *dialog* ilustrado na figura 3.7, onde é preciso informar os parâmetros pedidos. O prefixo IPv6 é necessário para a efetiva tradução de um endereço de rede IPv4 em IPv6. A *pool* IPv4 é necessária para traduzir um endereço IPv6 em um endereço IPv4 válido. Por fim, um endereço pertencente a *pool* IPv4 deve ser usado para endereçar o servidor *Tayga*, para fins de respostas a requisições e respostas ICMP.

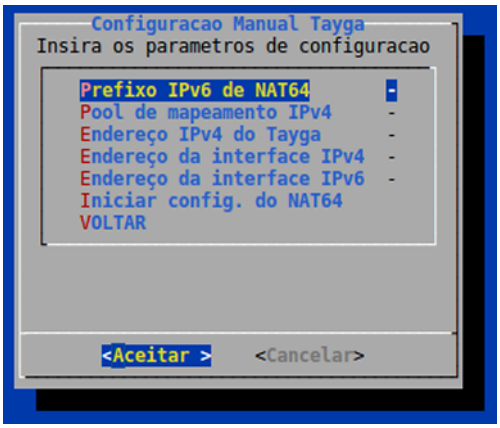

Figura 3.7: Menus da configuração do NAT64

No menu do NAT64, também são requeridos os endereços das interfaces de rede IPv4 e IPv6, logo, para a correta configuração do servidor *Tayga*, e consequentemente do serviço de tradução, a máquina usada deve possuir ao menos duas interfaces de rede, uma conectada à rede IPv6 e outra à internet ou qualquer outra rede que possua endereçamento IPv4.

Ao efetuar-se a configuração do NAT64, os parâmetros informados são usados para modificar o aquivo de configuração *Tayga*, *tayga.conf*, localizado no diretório:

```
/usr/local/etc/
```

Ao realizar-se a configuração do NAT64, o *script* também verifica se é necessária a realização de SNAT via *iptables* no *host*, e em caso positivo, este procedimento é realizado com as seguintes definições:

```
# iptables -t nat -A POSTROUTING -s <Pool de endereços IPv4 do Tayga> -o
<interface de rede> -j SNAT --to <Endereço da interface IPv4>
```

Dessa forma, após a tradução, todos os pacotes que sairão para a rede externa terão o mesmo endereço IPv4 da interface do roteador, que caso tenha sido atribuído por algum servidor DHCP será prontamente reconhecido e roteado sem necessidade de configurações adicionais em outros componentes de rede.

3.1.5 DNS64

O DNS64 é implementado utilizando o BIND9, e para o seu funcionamento em conjunto com o NAT64, é preciso informar o mesmo prefixo IPv6 utilizado na configuração do Tayga, figura 3.8. De forma a aumentar a segurança do servidor, também é preciso informar as redes IPv6 que tem permissão para consultas DNS. Deve-se também informar os endereços IP dos *forwarders* IPv4 e IPv6 utilizados para consultas DNS a servidores autoritativos externos.

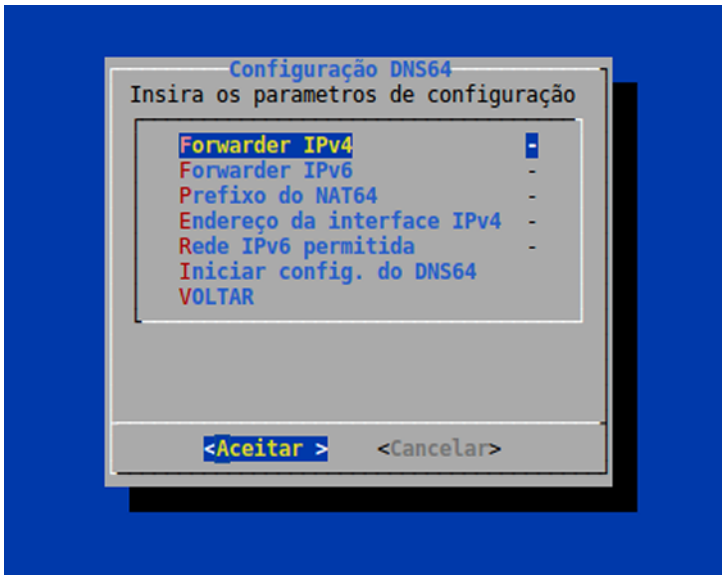

Figura 3.8: Menus da configuração do DNS64

Ao habilitar-se a configuração do DNS64, quarta opção do menu da figura 3.1, ainda é possível autorizar ou não a habilitação de DNSSEC, e se o DNS irá responder autoritativamente ao receber respostas *NXDOMAIN*. Outra funcionalidade extra é a autorização para que o DNS64 retorne apenas endereços IPv4 e traduza-os, na forma:

```
PREFIXO + ENDEREÇO IPv4
```

Após informados todos os parâmetros, esses são usados para alterar o aquivo de configuração BIND9, *named.conf.options*, localizado no diretório:

```
/etc/bind/
```

3.1.6 Exemplo de caso de uso - NAT64

A figura 3.9 mostra um diagrama de caso de uso para uma configuração de um *host* como servidor NAT64. Para efetuar tal operação, parte-se do pressuposto de que a máquina *Linux* esteja com o BIND9 instalado.

Para efetuar a correta configuração do ambiente de rede IPv4-IPv6, como explicado na seção 3.1.3, deve-se garantir que as configurações de prefixo IPv6, *pool* IPv4, endereços de rede IPv4 e IPv6, além do endereço do servidor Tayga sejam corretamente inseridas. Opcionalmente, pode-se ignorar a configuração do Tayga e voltar ao menu *dialog* anterior para demais configurações.

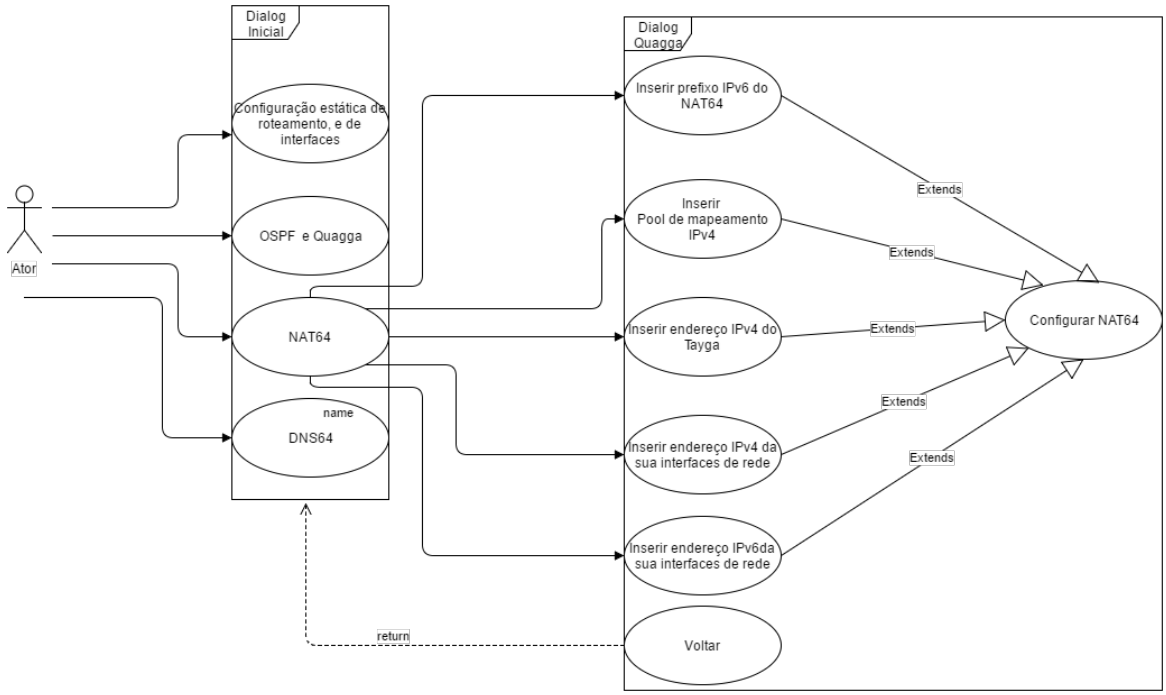

Figura 3.9: Caso de uso para configuração NAT64
[Autores: Paulo Henrique Leal / Lucas Ribeiro]

3.2 TOPOLOGIAS DE TESTE

De forma a avaliar o desempenho do NAT64 e do DNS64 foram propostas 3 topologias, sendo utilizados métodos variados para teste da mesmas. A figura 3.10 ilustra a topologia base utilizada para todos os testes, sendo os valores de endereço identificados em tabelas de cada subseção.

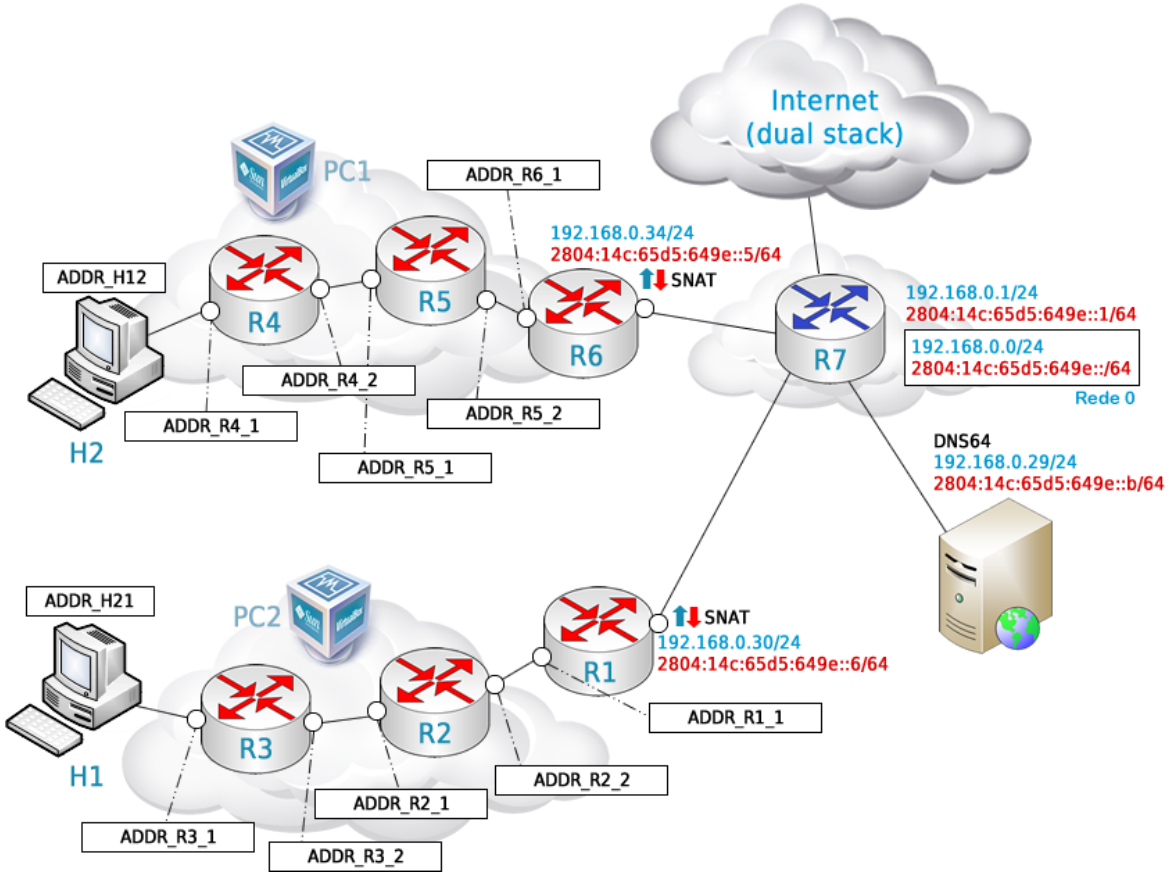

Figura 3.10: Topologia de testes de desempenho
[Autores: Paulo Henrique Leal / Lucas Ribeiro]

3.2.1 Topologia com endereçamento puramente IPv4

A topologia IPv4 foi proposta de forma a obter-se uma referência, pois o endereçamento IPv4 é comum em grande parte da rede mundial de computadores. A topologia da figura 3.10 foi composta por duas redes locais, uma rede 10.11.0.0/16 e outra rede 10.10.0.0/16. Cada rede foi simulada em um *desktop* via *Virtual Box*, onde cada rede possuía quatro máquinas virtuais *Linux* com o roteamento habilitado. As maquinas virtuais H1 e H2 foram designadas como *hosts* e foram montadas com somente uma interface de rede no modo *Rede Interna* no *Virtual Box*. Em contrapartida, os roteadores R2, R3, R4 e R5 contém duas interfaces de rede interna. Os roteadores R1 e R6, que são roteadores de borda, possuem uma interface no modo *bridge* para receber o endereço IP via DHCP do roteador R7 da rede externa, e suas respectivas interfaces de rede interna.

O endereçamento de cada interface das máquinas virtuais foi feito utilizando a ferramenta de configuração apresentada na seção anterior, que altera os arquivos de configuração do software *quagga* para realizar tal operação. O roteamento interno de cada rede foi efetuado utilizando a opção de configuração OSPF da ferramenta.

Para a devida comunicação entre as duas redes foram adicionadas rotas *default* de forma estática via ferramenta de configuração nas maquinas virtuais R4, R5 e R6 em direção

ao próximo salto, de forma que essas máquinas saibam encaminhar pacotes para a rede 10.10.0.0/16. Além do adição das rotas, é efetuado SNAT na maquina virtual R6 usando-se a ferramenta via dialog, de forma que qualquer máquina virtual da topologia PC1 se comunique com uma rede externa usando o endereço IPv4 que foi atribuído a interface *bridge* via DHCP pelo roteador *dual stack*. A tabela 3.1 especifica os endereços das interfaces de acordo com a topologia da figura 3.10.

Tabela 3.1: Especificação de endereços das interfaces para o cenário da figura 3.10

Variável de endereço	Endereço IP (CIDR /24)
ADDR_H12	10.11.3.2
ADDR_R4_1	10.11.3.1
ADDR_R4_2	10.11.2.2
ADDR_R5_1	10.11.2.1
ADDR_R5_2	10.11.1.2
ADDR_R6_1	10.11.1.1
ADDR_H21	10.10.3.2
ADDR_R3_1	10.10.3.1
ADDR_R3_2	10.10.2.2
ADDR_R2_1	10.10.2.1
ADDR_R2_2	10.10.1.2
ADDR_R1_1	10.10.1.1

3.2.2 Topologia com endereçamento puramente IPv6

A rede puramente IPv6 foi configurada como forma de obter-se uma base do desempenho dos testes em uma rede deste tipo. A topologia da figura 3.10 é composta por duas redes, a rede 2001:db8:1:0::/64, onde o quarto duocteto define as sub-redes. As configurações de cada máquina virtual foram as mesmas utilizadas em na topologia puramente IPv4. O roteamento interno da topologia de cada *desktop* foi efetuado usando-se o OSPFv6 do software *quagga*, por meio da ferramenta de configuração proposta.

Novamente para devida comunicação entre as duas redes foram adicionadas rotas *default* de forma estática via ferramenta de configuração nas máquinas virtuais R4, R5 e R6 em direção ao próximo salto, de forma que essas máquinas possam encaminhar pacotes para as redes das máquinas virtuais R1, R2, R3 e H2 e demais redes que venham a ser adicionadas.

Além do adição das rotas, é efetuado SNAT na máquina virtual R6 usando-se a ferramenta de configuração proposta, de forma que qualquer máquina virtual da topologia PC1 possa se comunicar com uma rede externa usando o endereço IPv6 que foi atribuído a interface *bridge* via DHCPv6 pelo roteador *dual stack*. A tabela 3.2 especifica os endereços das interfaces de acordo com a topologia da figura 3.10.

Tabela 3.2: Especificação de endereços das interfaces com endereçamento puramente IPv6 tendo como base a topologia da figura 3.10

Variável de endereço	Endereço IP (CIDR /64)
ADDR_H1	2001:DB8:1:4::2
ADDR_R4_1	2001:DB8:1:4::1
ADDR_R4_2	2001:DB8:1:5::2
ADDR_R5_1	2001:DB8:1:5::1
ADDR_R5_2	2001:DB8:1:6::2
ADDR_R6_1	2001:DB8:1:6::1
ADDR_H2	2001:DB8:1:3::2
ADDR_R3_1	2001:DB8:1:3::1
ADDR_R3_2	2001:DB8:1:2::2
ADDR_R2_1	2001:DB8:1:2::1
ADDR_R2_2	2001:DB8:1:1::2
ADDR_R1_1	2001:DB8:1:1::1

3.2.3 Topologia mista com NAT64 e DNS64

A topologia mista, tendo como base a figura 3.10, é o alvo de estudo deste trabalho, onde deseja-se verificar por meio dos testes a viabilidade da implementação NAT64 e do DNS64 e analisar se o desempenho é semelhante ou superior às redes puramente IPv6 e IPv4, tomando como base os parâmetros definidos na seção 3.3. A rede IPv6 onde está localizado o servidor Tayga com o NAT64 e o DNS64 é simulada no PC2 e possui 5 máquinas virtuais. De forma semelhante a rede puramente IPv6 da subseção 3.2.2, as máquinas virtuais H1 e H2 possuem apenas uma interface de Rede Interna. As máquinas virtuais R2, R3, R4 e R5 possuem duas interfaces de Rede Interna, e as máquinas virtuais R1 e R6 possuem uma interface em modo *bridge*, de forma a receber um endereço de rede IPv4 e IPv6 via DHCP ou DHCPv6 do roteador habilitado com *dual stack*.

A rede IPv4 teve a mesma configuração que o PC2 definido na seção 3.2.1, onde a interface de borda da máquina virtual em *bridge*, recebe um IP atribuído via DHCP pelo roteador. As configuração dos endereços IPv4 das interfaces de rede são especificados na tabela 3.3.

Tabela 3.3: Especificação de endereços das interfaces com endereçamento IPv4 tendo como base a topologia da figura 3.10

Variável de endereço	Endereço IP (CIDR /24)
ADDR_H12	10.10.3.2
ADDR_R4_1	10.10.3.1
ADDR_R4_2	10.10.2.2
ADDR_R5_1	10.10.2.1
ADDR_R5_2	10.10.1.2
ADDR_R6_1	10.10.1.1

As interfaces de rede foram configuradas com o *quagga* por meio da ferramenta de

configuração, assim como o roteamento interno das máquinas virtuais, que foi realizado usando OSPFv6. A configuração do servidor NAT64 é feito na máquina virtual de borda R1 utilizando a ferramenta de configuração, onde devem ser informadas interfaces de rede, prefixo IPv6 e a *pool* IPv4 que será usada tanto para dar um endereço ao servidor *quagga*, quanto para ser usada para mapear endereços das máquinas virtuais IPv6. Confirmando a opção de SNAT via *iptables*, todas as máquinas virtuais pertencentes a rede IPv6 têm seus endereços de rede da *pool* multiplexados no IPv4 atribuído a R1 via DHCP, gerando assim um mapeamento *stateful*. Na máquina virtual R1, como configuração adicional foi configurada rota para a rede 10.10.0.0/16 e rotas *default* nos roteadores R2 e R3 de forma que toda a topologia possa enviar pacotes com endereço do tipo PREFIXO::IPv4/96. Para se configurar o DNS64, basta informar os endereços da interface de rede *bridge* atribuídos pelo roteador e o prefixo utilizado no NAT64. A tabela 3.4 especifica os endereços das interfaces de acordo com a topologia da figura 3.10.

Tabela 3.4: Especificação de endereços das interfaces com endereçamento IPv6 tendo como base a topologia da figura 3.10

Variável de endereço	Endereço IP (CIDR /64)
ADDR_H21	2001:DB8:1:3::2
ADDR_R3_1	2001:DB8:1:3::1
ADDR_R3_2	2001:DB8:1:2::2
ADDR_R2_1	2001:DB8:1:2::1
ADDR_R2_2	2001:DB8:1:1::2
ADDR_R1_1	2001:DB8:1:1::1

3.2.4 Topologia para testes de conexão com a Internet

De forma a validar o funcionamento dos mecanismos de NAT64 e DNS64, propôs-se uma topologia específica, apresentada na figura 3.11, que permite a apresentação de um cenário onde uma rede puramente IPv6 encontra-se isolada, conectada a internet via roteador NAT64 e utilizando um servidor DNS64 como resolvedor de nomes. A diferença em relação a topologia da figura 3.10 dá-se pelo fato de que a configuração em ambas deve ser tratada de forma diferente por possuírem diferentes objetivos de análise.

Para confirmação do funcionamento da topologia apresentada na figura 3.11, aplicou-se a metodologia de testes de acesso à internet em redes puramente IPv6, puramente IPv4 e *Dual Stack* com o auxílio da ferramenta de *sniffer Wireshark*.

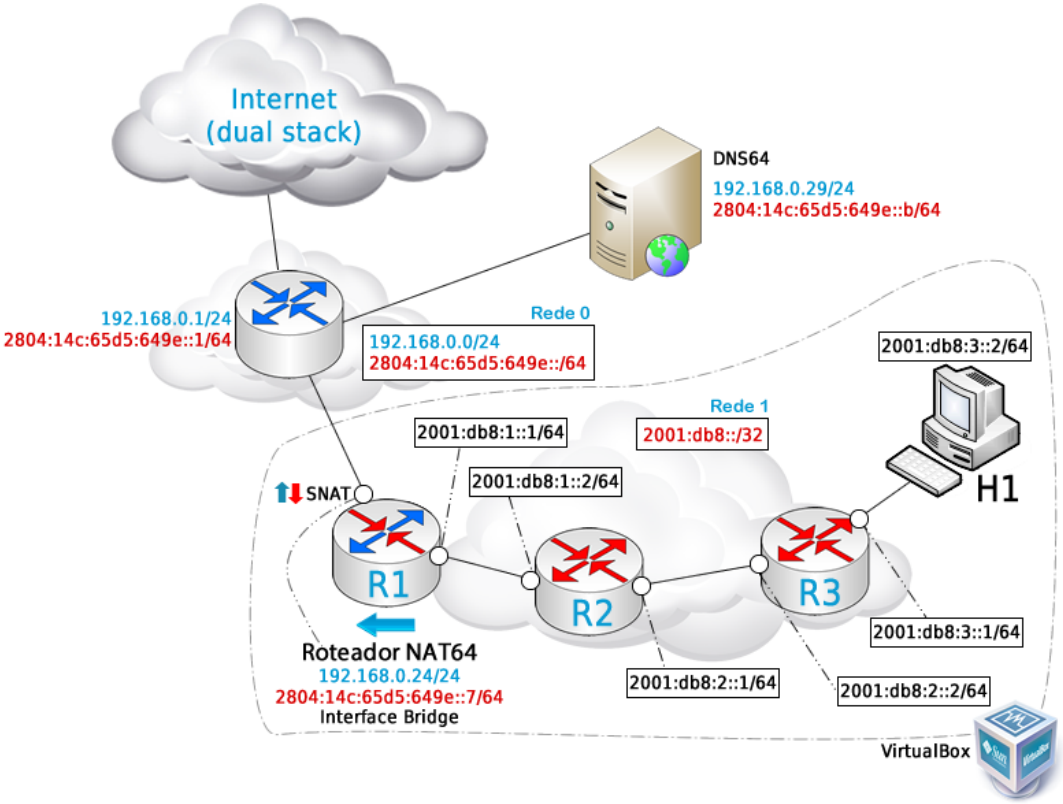

Figura 3.11: Topologia de testes de conexão com a internet
[Autores: Paulo Henrique Leal / Lucas Ribeiro]

3.3 TESTES DE DESEMPENHO DE REDE

3.3.1 Iperf

O *iperf3* é uma ferramenta cliente-servidor utilizada para medição de parâmetros em redes IP, medição essa que é feita trafegando dados do cliente para o servidor, neste caso, da máquina virtual H1 para a maquina virtual H2, referentes à figura 3.10. Os resultados do tráfego de dados são impressos no terminal, gerando assim flexibilidade de na exportação de dados para posterior análise.

No tráfego de dados na camada de transporte é possível selecionar os protocolos TCP ou UDP. Quando se utiliza o protocolo TCP, é possível medir largura de banda e tamanho do *MSS/MTU*. No tráfego UDP, pode-se criar várias *streams* de dados, cada uma com largura de banda especifica, além de ser possível medir a perda de pacotes e o *jitter*.

Para o teste de desempenho da topologia da figura 3.10, o Iperf foi utilizado para se medir o *troughput* e o *jitter*.

3.3.2 Troughput

Para o teste de *troughput* foram simulados nas três topologias o envio de um pacote FTP da maquina virtual H1 para a maquina virtual H2, figura 3.10. Para tal, utilizou-se o iperf3 com uma janela TCP de 8 KB (-w8K), típica de aplicações FTP, utilizando a porta 5001 padrão para comunicação TCP (-p5001). Os dados são transmitidos em uma única rajada (-P1), tendo o tamanho máximo de segmento *MSS* impresso no terminal (-M). A simulação ocorre durante 60 segundos, de forma que a cada 10 segundos é retornado o valor da largura de banda (-i10), com os dados representados na unidade de Kilo Bytes (-fK). Vale ressaltar que para as topologias que envolvem tráfego IPv6, é preciso se habilitar a opção de tráfego unicamente IPv6 (-6). O parâmetro –get-server-output é utilizado como forma de obter-se do lado cliente a saída gerada pelo servidor.

Tabela 3.5: Endereços IP de origem e destino utilizados nos testes de desempenho tendo como base a topologia da figura 3.10

Topologia	Endereço IP de Destino	Endereço IP de Destino
Puramente IPv4	10.11.3.2	10.10.3.2
Puramente IPv6	2001:DB8:1:4::2	2001:db8:1:3::2
NAT64 e DNS64	2001:db8:1:3::2	2001:db8:1:ffff::10.10.3.2

Para o lado cliente na topologia puramente IPv4, utilizando o endereço de destino da tabela 3.5, o seguinte comando foi executado:

```
# iperf3 -c <endereço da interface de destino> -P1 -i10 -m -p5001 \
-w8K -fK -t60 --get-server-output
```

Para a topologia com endereçamento puramente IPv6, seção 3.2.2, com a opção -6 ativada e endereço da interface de destino sumarizado na tabela 3.5, foram utilizados os parâmetros:

```
# iperf3 -6 -c <endereço da interface de destino> -P1 -i10 -m -p5001 \
-w8K -fK -t60 --get-server-output
```

Na topologia mista, a configuração utilizada foi semelhante a da topologia puramente IPv6, porém, dada a utilização do NAT64 com o prefixo 2001:db8:1:ffff::/96, o endereço de destino foi alterado de acordo com a tabela 3.5.

Do lado servidor da aplicação, ou seja, na maquina virtual H2 da topologia da figura 3.10, antes de iniciada a conexão, configura-se o servidor via iperf3, que segue a mesma lógica das configurações no cliente. Para a topologia puramente IPv4, o comando aplicado é:

```
# iperf3 -s -fK
```

Para as topologias puramente IPv6, seção 3.2.2, e mista, seção 3.2.3, o comando aplicado no servidor é:

```
# iperf3 -6 -s -fK
```

3.3.3 Jitter

Para o teste de *Jitter* utilizando o iperf3 para as três topologias, foram mantidos os mesmos endereços de destino da tabela 3.5 descritos no teste de *troughput*, entretanto, utilizou-se o protocolo UDP, que permite a medição do parâmetro de *jitter*.

Neste teste simula-se duas transmissões de vídeo (-P2) em diferentes qualidades/taxas de forma bidirecional utilizando codificação H.264 para vídeo e AAC para áudio. Utilizou-se o protocolo UDP (-u), taxa de transmissão de 960 kbps (vídeo 896 kbps + áudio 64 kbps), 1600 kbps (vídeo 1536 kbps + áudio 64 kbps) e 3200 kbps (vídeo 3072 kbps + áudio 128 kbps) (-b 960K, -b 1600K, -b 3200K, parâmetro <bitrate>), com a simulação ocorrendo por 60 segundos (-t 60), sendo os resultados coletados a cada 5 segundos (-i5) com formatação dos dados em *Kilo Bytes* (-fK).

Para a topologia puramente IPv4, do lado cliente, temos a configuração do iperf3:

```
# iperf3 -c <endereço da interface de destino> -u -b <bitrate>K -P2 \
-fK -i5 -t60 --get-server-output
```

Para a topologia puramente IPv6 e topologia mista com NAT64 ativo, utilizou-se o mesmo comando da topologia puramente IPv4, mas com a opção -6 ativa:

```
# iperf3 -6 -c <endereço da interface de destino> -u -b <bitrate>K -P2 \
-fK -i5 -t60 --get-server-output
```

Do lado servidor, para todas as topologias, a seguinte configuração do iperf3 é aplicada:

```
# iperf3 -s -i5 -fK
```

3.3.4 Latência

O teste de latência foi feito por meio de uma requisição ICMP, utilizando para isso o comando *ping*, que retorna os parâmetros de RTT mínimo, médio, máximo e o desvio dessas medidas. Os endereços de destino e origem utilizados no teste estão sumarizados na tabela 3.5.

4 RESULTADOS E ANÁLISE

4.1 ANÁLISE DE VERIFICAÇÃO DE CONEXÃO COM NAT64 E DNS64 VIA *WIRESHARK*

Para validar o funcionamento dos mecanismos de tradução do NAT64 e DNS64, utilizou-se a captura de pacotes via *wireshark*. Utilizou-se a topologia apresentada na figura 3.11, com os parâmetros e configuração detalhados no capítulo 3. Os resultados são mostrados nas subseções 4.1.1 a 4.1.4.

4.1.1 NAT64

A verificação do mecanismo de NAT64 foi realizada através da análise de pacotes IPv6 na interface interna da rede puramente IPv6 e os respectivos pacotes IPv4 traduzidos. As figuras 4.1 e 4.2 ilustram a composição desses pacotes.

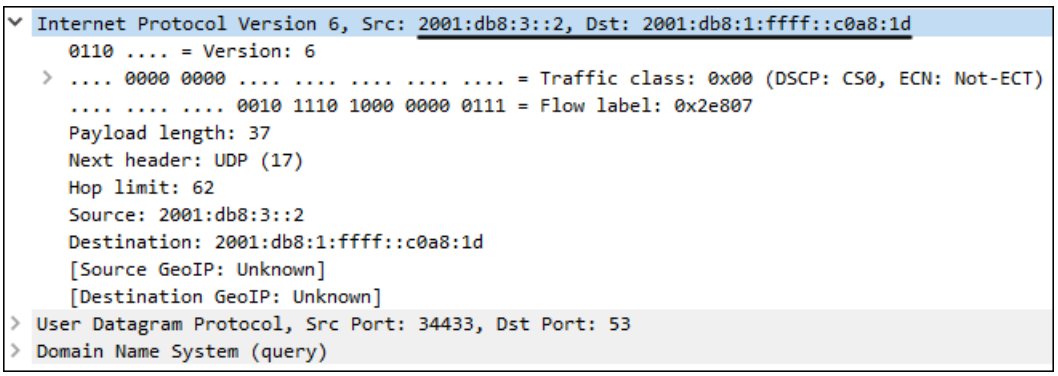

Figura 4.1: Pacote IPv6 capturado no roteador R1 da topologia da figura 3.11 via wireshark antes da tradução NAT64

No capítulo 2, seção 2.2.2.1, foi apresentada a tabela 2.10, que especifica a relação entre os campos do cabeçalho IPv6 (vide figura 2.5) e IPv4 (vide figura 2.2) em processos de tradução de acordo com a RFC 6145. Sendo assim, verifica-se a equidade entre o padrão estabelecido e o observado nos pacotes capturados. Faz-se necessária também a verificação da validade na tradução de endereços de origem e destino.

Em relação a tradução de cabeçalho, deve-se realizar uma análise campo a campo dos pacotes ilustrados nas figuras 4.1 e 4.2. O campo *Version* é alterado de 0110 (6) para 0100 (4). O campo IHL (tamanho de cabeçalho em palavras de 32 bits) tem valor 20 bytes, representado pelo valor 5, que indica a ausência de opções (tamanho mínimo). O campo *Differentiated Services Field*, equivalente ao campo *Type of Service* tem valor hexadecimal 0x00, sendo este uma cópia do campo *Traffic Class* do pacote IPv6. O campo *Total Length* tem tamanho 57, correspondente a soma do campo *Payload Length* do cabeçalho IPv6 com

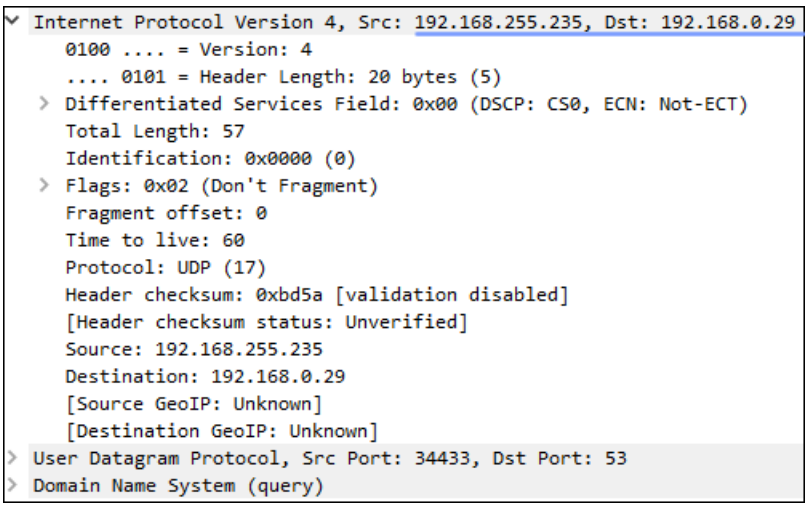

Figura 4.2: Pacote IPv4 capturado no roteador R1 da topologia da figura 3.11 via wireshark após a tradução NAT64

o campo IHL do IPv4 (37 + 20 = 57). O campo *Identification* tem valor hexadecimal 0x0000, indicando que o pacote não foi fragmentado. O campo *Flags* tem valor binário 010 (2), indicando os bits Reservado = 0, DF = 1 e MF = 0. O campo *Fragment Offset* é zerado, indicado ausência de fragmentação. O campo *Time to Live* tem valor 60, referente ao valor do campo IPv6 *Hop Limit* subtraído de 2 unidades, devido ao fato da passagem pelo túnel NAT64 consistir em um roteamento de dois saltos pelo kernel. Por fim, o campo protocolo tem valor 17, referente ao protocolo UDP, igualmente ao campo *Next Header* do cabeçalho IPv6, uma vez que o pacote capturado refere-se a uma requisição DNS. Dessa forma, se comparados os resultados apresentados ao que é especificado na tabela 2.10, observa-se que o processo de tradução de cabeçalho cumpriu em 100% os requisitos impostos pela RFC 6145, resultando em uma verificação de sucesso quanto a utilização do Tayga para processos de tradução NAT64.

Em relação a tradução de endereços, observa-se que o NAT64 realiza uma associação entre o endereço de origem IPv6 2001:db8:3::2 e o endereço IPv4 192.168.255.235. Essa associação é feita de acordo com o algoritmo do Tayga, e irá variar para diferentes endereços de origem dada a natureza *stateless* do tradutor. Em relação ao endereço de destino, observa-se a associação do endereço IPv6 2001:db8:1:ffff::c0a8:1d com o endereço IPv4 192.168.0.29. É importante observar que o endereço 2001:db8:1:ffff::c0a8:1d/96 corresponde ao equivalente endereço 2001:db8:1:ffff::192.168.0.29/96, ou seja, o endereço IPv4 é retirado dos últimos 32 bits do endereço IPv6 de destino, que consiste em um prefixo 2001:db8:1:ffff::/96 sufixado de um endereço IPv4, como estabelece as regras do NAT64. Dessa forma, verifica-se a validação também no processo de tradução de endereços de acordo com a teoria apresentada no capítulo 2, subseção 2.2.2.

4.1.2 DNS64

A verificação do mecanismo de DNS64 foi realizada através da análise de pacotes de requisição e respostas DNS nas interfaces de rede do servidor DNS64. Utilizando-se da topologia apresentada na figura 3.11, realiza-se conexão via web no host H1 a diferentes páginas da internet, sendo uma delas referente a um domínio de *pilha dupla* (www.nbc.com) e a outra a um domínio puramente IPv4 (www.ene.unb.br). Os resultados são apresentados nas figuras 4.3 e 4.4. É importante ressaltar que o endereço de origem das requisições DNS são alterados de 2001:db8:3::2 para 192.168.0.24, correspondente ao endereço da interface externa do roteador de borda, devido ao processo de SNAT realizado via *iptables* para possibilitar a conexão com a rede IPv4 externa.

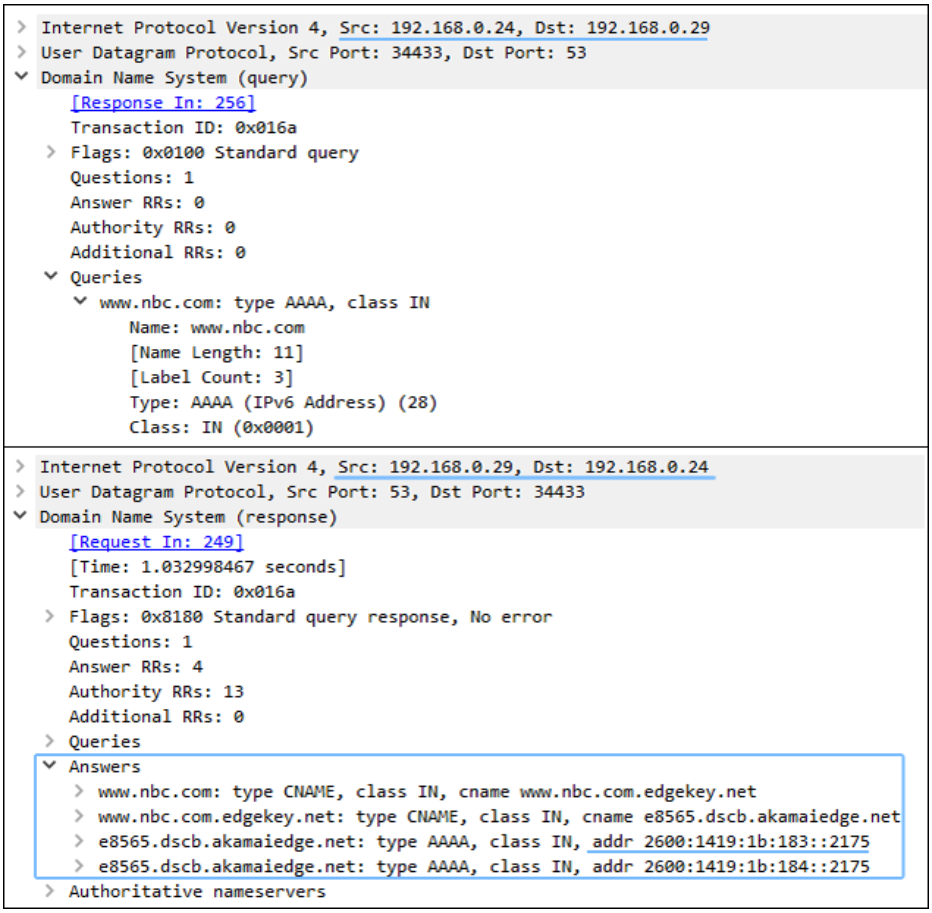

Figura 4.3: Requisição e resposta DNS para o domínio *www.nbc.com* capturado no roteador R1 da topologia da figura 3.11 via wireshark

Para conexão com o domínio web *www.nbc.com*, observa-se na figura 4.3 que primeiramente o endereço 192.168.0.24 realiza uma requisição ao servidor DNS (192.168.0.29) do tipo AAAA para o endereço web correspondente. Uma vez que o endereço IPv6 para o domínio solicitado (2600:1419:1b:183::2175) é recebido do servidor DNS autoritativo (processo não ilustrado na imagem por questões de simplificação), este é diretamente retornado ao host de origem da requisição, como observa-se no campo *Answers*. Este procedimento verifica e comprova o comportamento do DNS64 para a existência de registros AAAA nativos

do domínio solicitado, uma vez que este foi diretamente retornado ao host de origem.

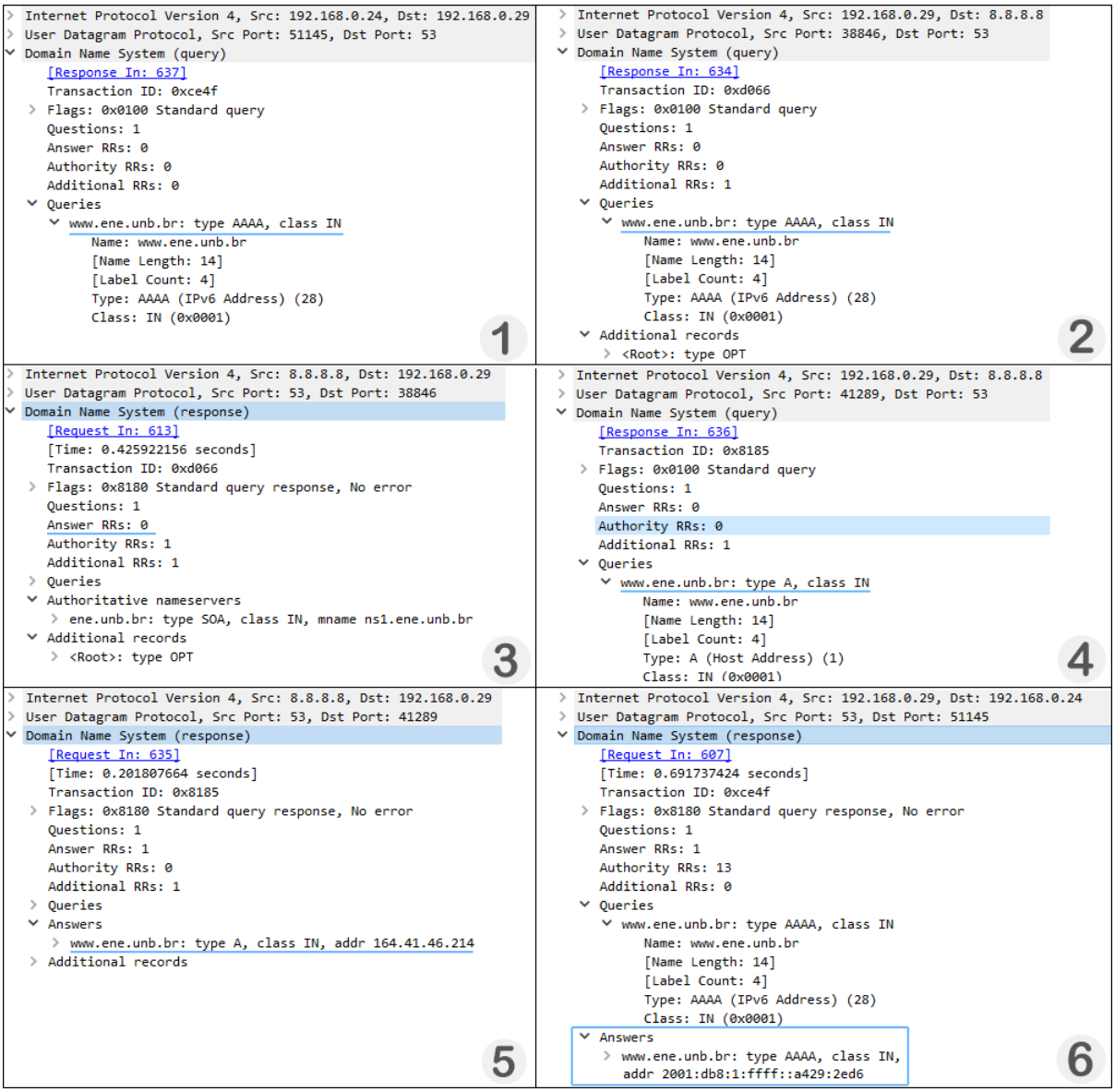

Figura 4.4: Requisição e resposta DNS para o domínio *www.ene.unb.br* capturado no roteador R1 da topologia da figura 3.11 via wireshark

Para o domínio puramente IPv4 *www.ene.unb.br*, verifica-se as requisições e respostas ilustradas na figura 4.4. Como observa-se na figura, a resposta DNS final é obtida após 6 passos, assim como foi ilustrado no exemplo do esquema da figura 2.15. Vale ressaltar novamente que o processo de SNAT realizado na borda do roteador NAT64 mascara os endereços IPv6 em um endereço IPv4 192.168.0.24. O passo 1 consiste na requisição de um registro do tipo AAAA do host H1 ao servidor DNS64 para o domínio *www.ene.unb.br*. O passo 2 consiste na requisição do servidor DNS64 (de endereço 192.168.0.29) do registro AAAA para o domínio requerido pelo host H1 a um servidor DNS autoritativo externo, que neste caso, tem endereço 8.8.8.8 (*Google DNS*). O passo 3 consiste na resposta de ausência de registro AAAA recebida pelo DNS64 advinda do servidor DNS autorita-

tivo. O passo 4 consiste na requisição de um registro do tipo A pelo DNS64 para o DNS autoritativo. O passo 5 consiste no recebimento da resposta pelo DNS64 com o registro A solicitado, 164.41.46.214. Por fim, no passo 6, o servidor DNS64 retorna ao endereço requisitante (192.168.0.24) um registro AAAA no formato PREFIXO::IPv4, de valor 2001:db8:1:ffff::a429:2ed6 ou 2001:db8:1:ffff::164.41.46.214. Dessa forma, se comparados os resultados apresentados ao que é especificado na figura 2.15, observa-se que o processo de tradução de endereço via DNS cumpriu em 100% os requisitos impostos pela RFC 6147, resultando em uma verificação de sucesso quanto a utilização do BIND9 para processos de requisição DNS64.

4.1.3 Teste de Conexão IPv6 com rede puramente IPv4

Após a verificação da eficácia dos mecanismos de NAT64 e DNS64, é possível então a verificação de uma conexão da rede puramente IPv6 implementada com uma rede externa IPv4. Isso pode ser comprovado através da captura de pacotes após o acesso via web ao domínio *www.ene.unb.br*, que localiza-se em uma rede puramente IPv4.

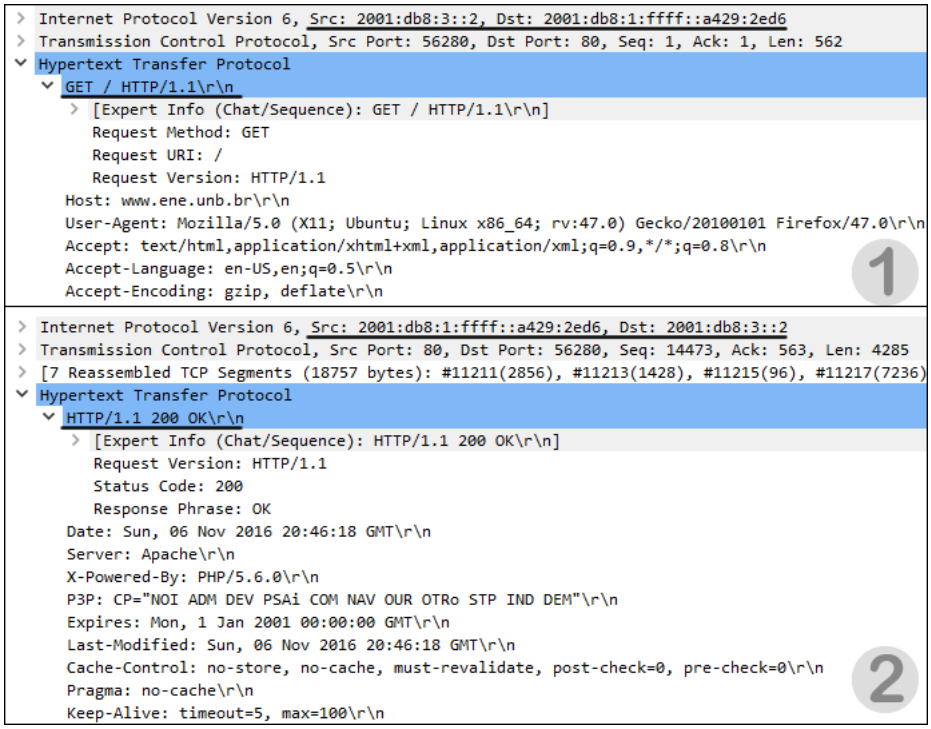

Figura 4.5: Requisições e respostas HTTP para o domínio *www.ene.unb.br* capturadas via wireshark no host H1 da topologia da figura 3.11

Para confirmar a possibilidade de conexão com a rede puramente IPv4 externa, analisa-se um pacote de requisição e sua respectiva resposta. Como ilustra a figura 4.5, o host H1 de endereço 2001:db8:3::2, ao solicitar uma página web via HTTP com uma requisição GET, recebe com sucesso a resposta OK 200 do protocolo HTTP. Observa-se que o endereço de destino da requisição e de origem da resposta são exatamente os endereços obtidos na

consulta DNS ilustrada na figura 4.4. Sendo assim, confirma-se a possibilidade de conexão com uma rede puramente IPv4 a partir de uma rede puramente IPv6 utilizando-se NAT64 e DNS64.

4.1.4 Teste de Conexão IPv6 com rede puramente IPv6

Uma vez que o núcleo da rede utilizada para testes é do tipo *Dual Stack*, é possível também a verificação da comunicação entre duas redes puramente IPv6. Desta vez, o NAT64 e o DNS64 irão operar como roteadores IPv6 padrão, sem a necessidade de tradução ou adição de mecanismos extras. Como forma de verificação, analisa-se a captura de pacotes com destino ao domínio IPv6-only *www.v6.facebook.com*.

Source	Destination	Protocol	Length	Info
2001:db8:3::2	2a03:2880:f005:8:face:b00c:0:1	TCP	94	56020→443 [SYN] Seq=0
2a03:2880:f005:8:face:b00c:0:1	2001:db8:3::2	TCP	94	443→56020 [SYN, ACK] Seq=0 Ack=1
2001:db8:3::2	2a03:2880:f005:8:face:b00c:0:1	TCP	86	56020→443 [ACK] Seq=1 Ack=1

Figura 4.6: Handshake TCP para o domínio *www.v6.facebook.com* capturado via wireshark no host H1 da topologia da figura 3.11

Para confirmar a possibilidade de conexão com a rede puramente IPv6 externa, analisa-se um pacote de requisição e suas respectivas resposta. Como ilustra a figura 4.6, o host H1 de endereço 2001:db8:3::2, ao necessitar conectar-se via protocolo TCP com o domínio *www.v6.facebook.com*, de endereço IPv6 2a03:2880:3010:df03:face:b00c:0:1, realiza com sucesso o processo de *Three-way Handshake* padrão do protocolo TCP. O endereço de destino da requisição e de origem da resposta são obtidos via DNS64. Sendo assim, confirma-se a possibilidade de conexão com uma rede puramente IPv6 a partir de uma rede também puramente IPv6 utilizando-se roteadores NAT64 e servidores DNS64 que operam com funcionamento padrão.

4.2 ANÁLISE DE DESEMPENHO DO NAT64 E DNS64

4.2.1 Teste de latência via ICMP Echo Request (PING)

A partir da topologia apresentada na figura 3.10, mediu-se o tempo médio que um pacote de 64 bytes leva para trafegar do host H1 ao host H2, através do comando PING, tempo este equivalente à latência da rede. Foram realizados testes para todas as topologias apresentadas nas seções 3.2.1 a 3.2.3. Os resultados obtidos são ilustrados no gráfico da figura 4.7 e especificados na tabela 4.1.

O gráfico da figura 4.7 ilustra a variação no valor médio de latência a cada 10 amostras obtidas em um total de 100. Teoricamente, a simplificação do cabeçalho IPv6 e seu tamanho fixo em relação ao cabeçalho IPv4 contribui em uma diminuição de processamento por parte dos roteadores e, consequentemente, na diminuição da latência média em redes

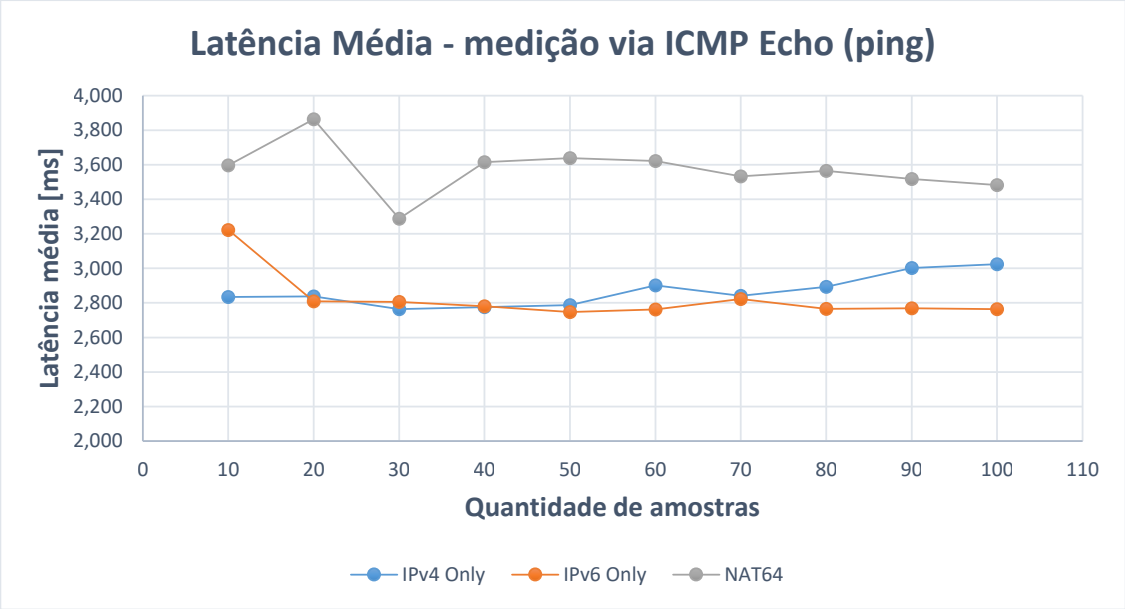

Figura 4.7: Gráfico de média de latência referentes à figura da topologia 3.10 com endereçamento IPv4-Only, IPv6-Only e tradução NAT64

IPv6. Entretanto, observa-se que, em média, a latência nas redes IPv4-Only e IPv6-Only apresentam-se próximas graficamente, sendo o segundo caso cerca de 1,5% inferior. Este resultado implica na inferência de que a simplificação e o tamanho fixo do cabeçalho IPv6 em relação ao IPv4 não apresentam uma melhora significativa em termos de latência na rede para o cenário de testes apresentado.

De acordo com os resultados obtidos no gráfico e na tabela, observa-se uma diferença perceptível na média de latência para a topologia NAT64. A topologia apresentou um acréscimo médio de 24,6% (0,706 milissegundos) em relação à topologia puramente IPv4 e de 26,4% (0,747 milissegundos) em relação à puramente IPv6. Ainda que em termos de porcentagem os valores obtidos possam ser considerados significativos, é importante

Tabela 4.1: Média de latência para as topologias IPv4-Only, IPv6-Only e NAT64

Número de amostras	IPv4 Only (ms)	IPv6 Only (ms)	NAT64 (ms)
10	2,835	3,223	3,596
20	2,838	2,810	3,864
30	2,764	2,807	3,288
40	2,776	2,780	3,615
50	2,787	2,747	3,638
60	2,902	2,763	3,622
70	2,840	2,823	3,532
80	2,892	2,766	3,564
90	3,003	2,770	3,518
100	3,024	2,764	3,483
Média	**2,866**	**2,825**	**3,572**

ressaltar que a diferença em *milissegundos* entre as médias obtidas não causam grande impacto em aplicações de rede usuais, como acesso *web* e aplicações em tempo real, que suportam latências de até 200 *ms*[36]. Sendo assim, pode-se considerar que o processo de tradução de cabeçalho e endereço realizado pelo NAT64 não é prejudicial ao desempenho geral da rede em termos de adição de latência / atrasos.

4.2.2 Teste de throughput TCP

A partir da topologia apresentada na figura 3.10, obtém-se o *throughput* médio de uma aplicação tipicamente FTP do host H1 ao host H2, através do software *iperf*, como indicado no capítulo 3. Foram realizados testes para todas as topologias apresentadas nas seções 3.2.1 a 3.2.3. Os resultados obtidos são ilustrados no gráfico da figura 4.8 e especificados na tabela 4.2.

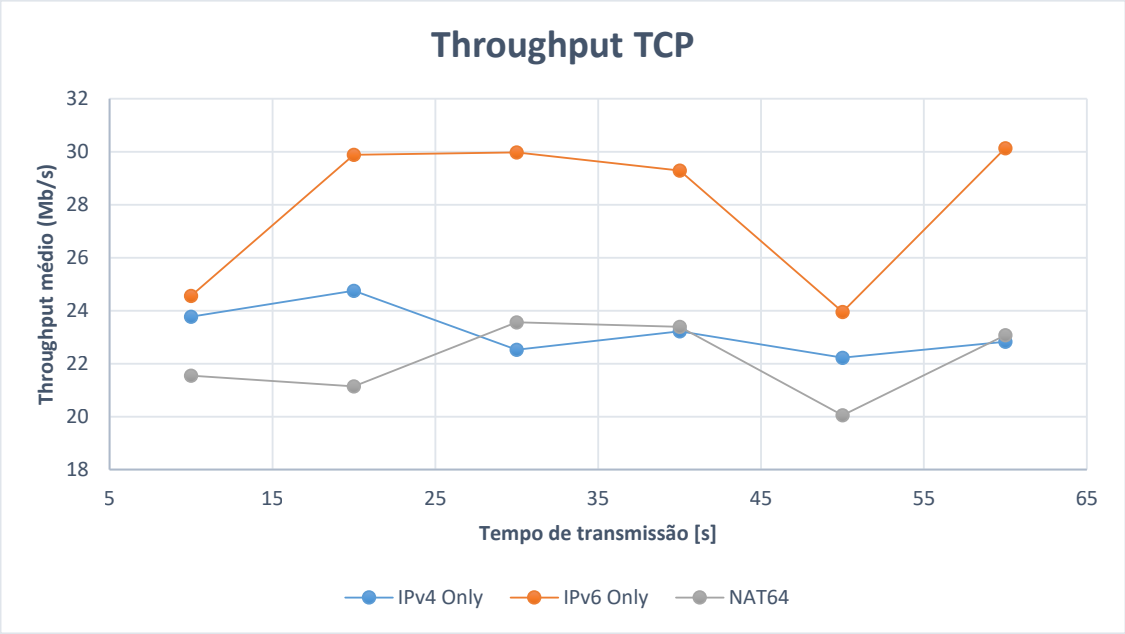

Figura 4.8: Gráfico de média de throughput referentes à figura da topologia 3.10 com endereçamento IPv4-Only, IPv6-Only e tradução NAT64

Tabela 4.2: Valores de média de throughput para as topologias IPv4-Only, IPv6-Only e NAT64

Tempo (seg.)	IPv4 Only (Mb/s)	IPv6 Only (Mb/s)	NAT64 (Mb/s)
10	23,768	24,561	21,544
20	24,752	29,882	21,144
30	22,523	29,968	23,562
40	23,216	29,288	23,392
50	22,224	23,952	20,048
60	22,824	30,136	23,072
Média	**23,217**	**27,964**	**22,127**

O gráfico da figura 4.8 ilustra a variação no *throughput* das redes implementadas para

a aplicação FTP (TCP) em um período de 60 segundos. As amostras foram tomadas em períodos de 10 segundos, onde cada amostra representa o *throughput* médio relativo ao último período de 10 segundos.

Os resultados obtidos no gráfico da figura 4.8, referentes à tabela 4.2, mostram que para uma rede puramente IPv6, o *throughput* tem valor superior às demais. Em média, em relação a rede IPv4 Only, a rede IPv6 apresenta aumento de 20,4% de *throughput*. Como explanado no capítulo 2, o datagrama IPv6 apresenta um acréscimo significativo em relação a quantidade de bytes em seu cabeçalho, o que justifica uma quantidade superior de dados sendo trafegada pela rede para redes que utilizam apenas o protocolo IPv6.

Para a topologia que utiliza o NAT64, nota-se, em média, um decréscimo de 4,7% em relação à rede IPv4-Only e um decréscimo de 21% em relação à rede IPv6-Only em termos de *throughput* na rede. Uma vez que o NAT64 realiza uma tradução do cabeçalho IPv6 em um IPv4, há uma diminuição na quantidade média de bits trafegados pela rede na porção IPv4 desta, o que justifica a queda de *throughput* na rede como um todo. Este resultado comprova a aplicabilidade do NAT64 em redes IPv4-IPv6 com alto fluxo de tráfego TCP, uma vez que este mecanismo causa impacto mínimo em relação às redes IPv4 existentes e pretende-se utilizá-lo como forma de coexistência com estas.

4.2.3 Teste de variação de atraso (Jitter) em aplicações UDP

A partir da topologia apresentada na figura 3.10, obtém-se o *Jitter* médio de uma aplicação tipicamente de áudio e vídeo que utiliza o protocolo UDP do host H1 ao host H2, através do software *iperf*. Foram realizados diversos testes para todas as topologias apresentadas nas seções 3.2.1 a 3.2.3. Os resultados obtidos são especificados nas subseções seguintes.

4.2.3.1 Vídeo em codificação H.264, resolução de 360p e faixa de áudio AAC de taxa 64 kbps

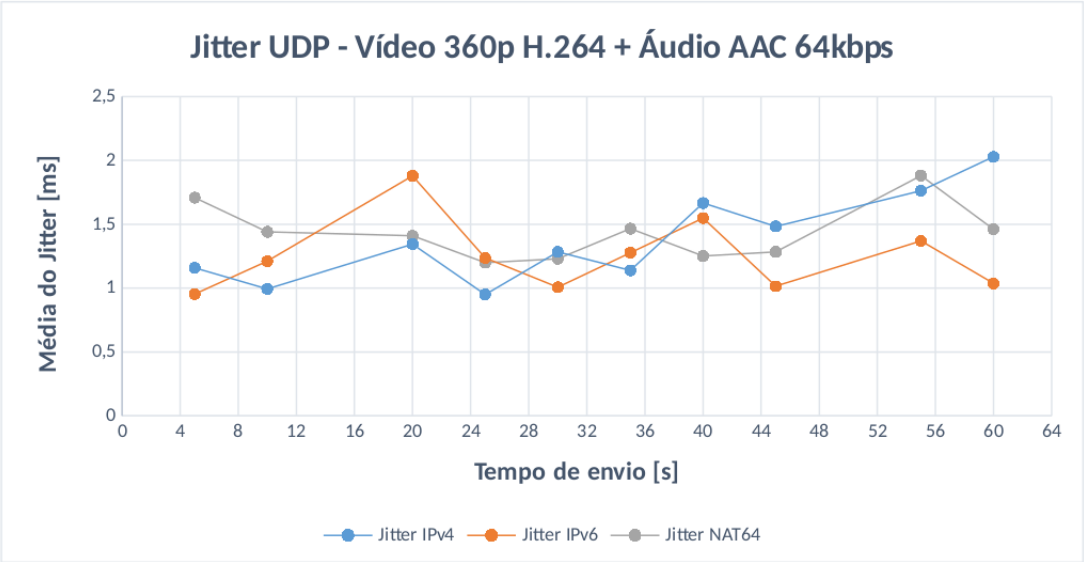

Figura 4.9: Gráfico de média na variação de atraso referentes à figura da topologia 3.10 com endereçamento IPv4-Only, IPv6-Only e tradução NAT64 em aplicação de áudio e vídeo de alta qualidade

Tabela 4.3: Valores de média na variação de atraso para as topologias IPv4-Only, IPv6-Only e NAT64 em aplicação de áudio e vídeo de baixa qualidade

Tempo (seg.)	IPv4 Only (ms)	IPv6 Only (ms)	NAT64 (ms)
5	1,159	0,954	1,709
10	0,994	1,211	1,441
20	1,345	1,88	1,41
25	0,951	1,237	1,199
30	1,284	1,008	1,23
35	1,139	1,278	1,466
40	1,667	1,549	1,252
45	1,484	1,015	1,284
55	1,763	1,37	1,882
60	2,029	1,037	1,462
Média	**1,382**	**1,254**	**1,434**

4.2.3.2 Vídeo em codificação H.264, resolução de 480p e faixa de áudio AAC de taxa 128 kbps

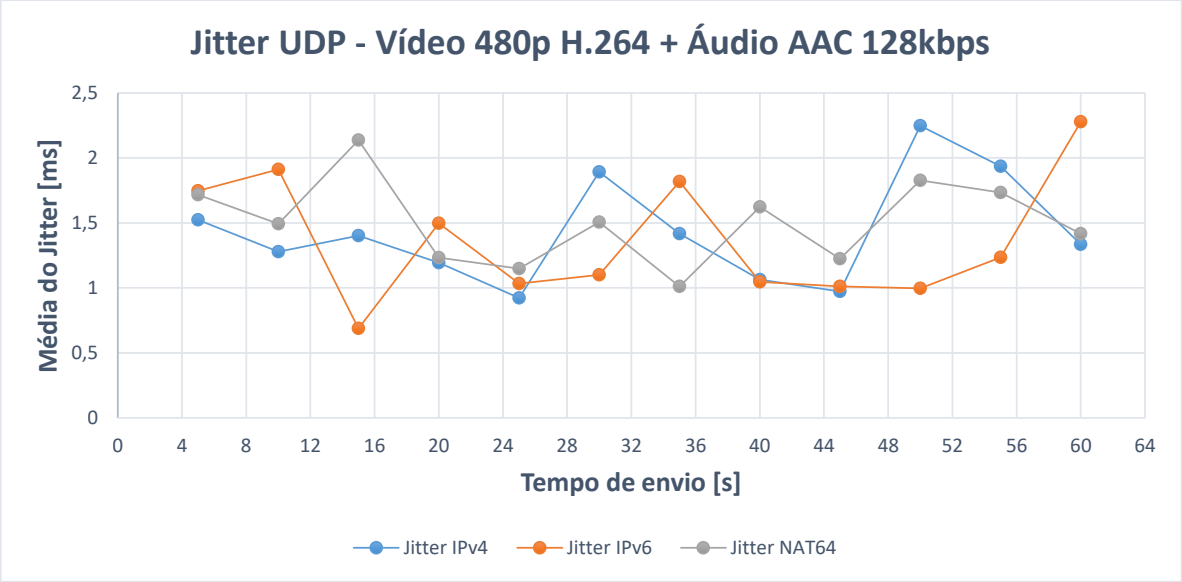

Figura 4.10: Gráfico de média na variação de atraso referentes à figura da topologia 3.10 com endereçamento IPv4-Only, IPv6-Only e tradução NAT64 em aplicação de áudio e vídeo de alta qualidade

Tabela 4.4: Valores de média na variação de atraso para as topologias IPv4-Only, IPv6-Only e NAT64 em aplicação de áudio e vídeo de média qualidade

Tempo (seg.)	IPv4 Only (ms)	IPv6 Only (ms)	NAT64 (ms)
5	1,524	1,748	1,717
10	1,278	1,911	1,493
15	1,403	0,688	2,137
20	1,195	1,497	1,233
25	0,923	1,034	1,148
30	1,893	1,101	1,506
35	1,418	1,819	1,011
40	1,064	1,047	1,624
45	0,973	1,011	1,224
50	2,248	0,997	1,828
55	1,936	1,236	1,735
60	1,334	2,278	1,418
Média	**1,432**	**1,364**	**1,506**

4.2.3.3 Vídeo em codificação H.264, resolução de 720p e faixa de áudio AAC de taxa 128 kbps

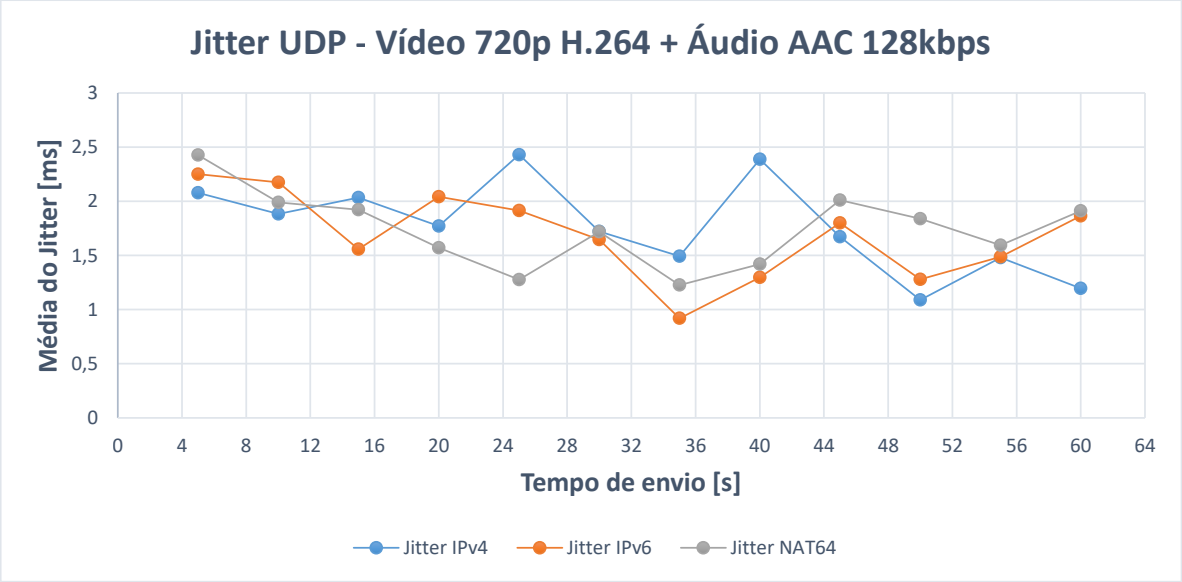

Figura 4.11: Gráfico de média na variação de atraso referentes à figura da topologia 3.10 com endereçamento IPv4-Only, IPv6-Only e tradução NAT64 em aplicação de áudio e vídeo de alta qualidade

Tabela 4.5: Valores de média na variação de atraso para as topologias IPv4-Only, IPv6-Only e NAT64 em aplicação de áudio e vídeo de alta qualidade

Tempo (seg.)	IPv4 Only (ms)	IPv6 Only (ms)	NAT64 (ms)
5	2,079	2,251	2,427
10	1,882	2,176	1,991
15	2,034	1,558	1,923
20	1,772	2,041	1,57
25	2,431	1,915	1,278
30	1,723	1,646	1,72
35	1,493	0,919	1,226
40	2,389	1,297	1,419
45	1,672	1,801	2,012
50	1,089	1,279	1,839
55	1,48	1,488	1,594
60	1,197	1,865	1,912
Média	**1,770**	**1,686**	**1,743**

Os gráficos das figuras 4.9, 4.10 e 4.11 ilustra a variação no *jitter* das redes implementadas para a aplicação de áudio e vídeo UDP em um período de 60 segundos. As amostras foram tomadas em períodos de 10 segundos, onde cada amostra representa a variação no atraso relativo ao último período de 10 segundos, ou seja, a diferença entre o maior e menor atraso no respectivo período de tempo. Deve-se ressaltar, que para o gráfico da figura 4.9, alguns períodos de tempo foram retirados das amostras por apresentarem-se discrepantes em relação aos resultados gerais, o que ocorre devido a aleatoriedades e demais instabilidades das máquinas virtuais e aplicações utilizadas.

Os resultados obtidos no gráfico da figura 4.9, referentes à tabela 4.3, mostram que

para uma rede puramente IPv6 com tráfego de áudio e vídeo de baixa qualidade, o jitter apresentou valor inferior às demais. Em média, em relação a rede IPv4 Only, a rede IPv6 apresenta decréscimo de 9,2% no *jitter*. Para aplicações de média e alta qualidade, o padrão repete-se e obtém-se um decréscimo de 4,8% para o mesmo caso. Como explanado no capítulo 2, o datagrama IPv6 apresenta uma simplificação em seu cabeçalho, o que agiliza o processo de roteamento pelos roteadores na rede e justifica a menor variação no atraso nos fluxos de pacotes. Justifica-se uma diminuição do *jitter* nas redes IPv6 também por conta da ausência de necessidade dos roteadores em realizar fragmentação de pacotes e consequente análise a cada salto na rede, uma vez que este processo é feito exclusivamente pelos hosts finais.

Para a topologia que utiliza o NAT64, nota-se, na média, um acréscimo de 3,8% em relação à rede IPv4-Only e um acréscimo de 14,3% em relação à rede IPv6-Only em termos de *jitter* na rede para aplicações de áudio e vídeo de baixa qualidade. Para aplicações de média qualidade, o aumento em relação à rede IPv4 foi de 5,2%, e de 10,4% em relação à rede IPv6. Por fim, para aplicação de alta qualidade, obtém-se um decréscimo de 1,7% em relação à rede IPv4 e um aumento de 3,4% em relação à rede IPv6. Uma vez que o NAT64 realiza uma tradução do cabeçalho IPv6 em um IPv4, espera-se que este processo acrescente alguma variação no atraso no processamento de pacotes no roteador de borda que o implementa, o que comprova-se pelo acréscimo de *jitter* em relação às demais topologias. Este resultado comprova a aplicabilidade do NAT64 em redes IPv4-IPv6 com alto fluxo paralelo de áudio e vídeo UDP em baixa qualidade, uma vez que este mecanismo causa impacto mínimo em relação às redes IPv4 existentes e pretende-se utilizá-lo como forma de coexistência com estas.

5 CONCLUSÃO E TRABALHOS FUTUROS

5.1 CONCLUSÕES GERAIS

Ao longo deste trabalho foram analisados diferentes aspectos que envolvem ambientes de redes IP. Primeiramente, verificou-se a possibilidade da construção de ambientes de redes através de softwares de virtualização e compartilhamento de hardware para análise de tráfego e demais propriedades relacionadas ao tema. Posteriormente, analisou-se, através da implementação de um ambiente de rede IPv4-IPv6 por meio de scripts na linguagem *bash*, a possibilidade da utilização do mecanismo de NAT64/DNS64 em redes diversas e o impacto causado por este mecanismo em diferentes tipos de tráfego multimídia.

Em relação à proposta de construção de uma ferramenta que utiliza softwares livres para a configuração de ambientes de rede virtualizados, considera-se que este objetivo foi devidamente alcançado, uma vez que a comunicação entre os componentes de rede das topologias propostas no capítulo 3 foi realizada com sucesso em seus diversos aspectos, como roteamento estático e dinâmico, configuração de interfaces, NAT64, DNS64 e conexão com a internet. Esse resultado comprova a eficácia e eficiência da virtualização de componentes de rede, uma vez que foi possível utilizar-se de reais componentes de rede para as simulações realizadas, que podem vir a gerar resultados mais precisos em termos de análise de tráfego, roteamento, etc., em relação à simuladores diversos.

Em relação à análise de aplicações multimídia em redes IPv4-IPv6 que utilizam como forma de coexistência o mecanismo de NAT64/DNS64 e a possibilidade da utilização destes como forma de prover acesso às redes externas em ilhas puramente IPv6, consideram-se atingidos os objetivos propostos. Como resultado, observa-se que a utilização do NAT64/DNS64 em redes que utilizam diferentes versões do protocolo IP causa impacto pouco significativo em termos de utilização de banda, jitter e latência se comparados a redes que utilizam apenas um dos protocolos. A principal diferença notada em redes NAT64/DNS64 está na latência experimentada por estas, que ocorre devido ao processo de tradução dos cabeçalhos IPv6 em IPv4 e vice-versa. Não foram observados problemas relacionados ao acesso à internet em domínios IPv4 ou IPv6 por hosts em redes puramente IPv6 isoladas. De forma geral, observa-se que o NAT64/DNS64 mostra-se como uma alternativa confiável e eficaz para ser utilizado em ambientes de redes diversos.

5.2 TRABALHOS FUTUROS

- Implementar políticas de roteamento IPv6 baseadas na utilização do campo Flow Label para análise de QoS nos ambientes de redes apresentados.

- Implementar diferentes algoritmos de roteamento em conjunto com o NAT64/DNS64, como o AODV, BGP, etc.
- Aprimorar os scripts implementados adicionando funções de execução remota, tunelamento e demais funcionalidades extras.

REFERÊNCIAS BIBLIOGRÁFICAS

1 RéPáS, S. Application compatibility of the nat64 ipv6 transition technology. v. 1, p. 1–7, 2015.

2 BARAYUGA, V. J. D. Packet level tcp performance of nat44, nat64 and ipv6 using iperf in the context of ipv6 migration. v. 1, p. 1–3, 2015.

3 JIN, R. Provide ipv4 service using pure ipv6 servers with stateless nat64 translator. v. 1, p. 14–23, 2014.

4 LENCSE, G. Performance analysis and comparison of the tayga and of the pf nat64 implementations. v. 1, p. 71–76, 2013.

5 LENCSE, G. Performance analysis and comparison of different dns64 implementations for linux, openbsd and freebsd. v. 1, p. 877–844, 2013.

6 HODZIC, E. Ipv4/ipv6 transition using dns64/nat64: Deployment issues. v. 1, p. 1–6, 2012.

7 KUROSE, J. F. *Redes de computadores e a Internet*. 5ª. ed. [S.l.: s.n.], 2009.

8 POSTEL, J. *RFC 791 - INTERNET PROTOCOL*. 1981. [Acessado em: 14/09/2016]. Disponível em: <https://tools.ietf.org/html/rfc791>.

9 KOZIEROK, C. M. *IP Datagram General Format*. 2005. [Acessado em: 16/09/2016]. Disponível em: <http://www.tcpipguide.com/free/t_IPDatagramGeneralFormat.htm>.

10 KOZIEROK, C. M. *The TCP/IP Guide*. 1ª. ed. [S.l.: s.n.], 2005.

11 POSTEL, J. *RFC 790 - ASSIGNED NUMBERS*. 1981. [Acessado em: 17/09/2016]. Disponível em: <https://tools.ietf.org/html/rfc790>.

12 REKHTER, Y. *RFC 1918 - Address Allocation for Private Internets*. 1996. [Acessado em: 16/09/2016]. Disponível em: <https://tools.ietf.org/html/rfc1918>.

13 POSTEL, J. *RFC 792 - INTERNET CONTROL MESSAGE PROTOCOL*. 1981. [Acessado em: 30/10/2016]. Disponível em: <https://tools.ietf.org/html/rfc792>.

14 DEERING, S. *RFC 2460 - Internet Protocol, Version 6 (IPv6) Specification*. 1998. [Acessado em: 16/09/2016]. Disponível em: <https://tools.ietf.org/html/rfc2460>.

15 ENACHE, D. A study of the technology transition from ipv4 to ipv6. v. 1, p. 1–6, 2016.

16 CISCO. *IPv6 Extension Headers Review and Considerations*. 2006. [Acessado em: 22/09/2016]. Disponível em: <http://www.cisco.com/en/US/technologies/tk648/tk872/technologies_white_paper0900aecd8054d37d.html>.

17 NIC.BR. *Endereçamento IPv6*. 2012. [Acessado em: 28/09/2016]. Disponível em: <http://ipv6.br/post/enderecamento/>.

18 DAS, K. *ICMPv6 - Tech Details Advantages*. [Acessado em: 25/10/2016]. Disponível em: <http://ipv6.com/articles/general/ICMPv6.htm>.

19 NRO. *Free Pool of IPv4 Address Space Depleted*. 2011. [Acessado em: 26/09/2016]. Disponível em: <https://www.nro.net/news/ipv4-free-pool-depleted>.

20 IPV6MATRIX. *IPv6 adoption around the Globe*. 2016. [Acessado em: 26/09/2016]. Disponível em: <http://www.ipv6matrix.org/>.

21 DAS, K. *IPv6 Transition Technologies*. [Acessado em: 06/10/2016]. Disponível em: <http://ipv6.com/articles/gateways/IPv6-Tunnelling.htm>.

22 IETF. *RFC 4213 - Basic Transition Mechanisms for IPv6 Hosts and Routers*. 2010. [Acessado em: 05/10/2016]. Disponível em: <https://tools.ietf.org/html/rfc4213>.

23 LAHEY, K. *RFC 2993 - Architectural Implications of NAT*. 2000. [Acessado em: 26/09/2016]. Disponível em: <https://tools.ietf.org/html/rfc2923>.

24 FLOYD, S. *RFC 4782 - Behavioral Requirements for Unicast UDP*. 2007. [Acessado em: 26/09/2016]. Disponível em: <https://tools.ietf.org/html/rfc4782>.

25 LI, X. *RFC 6145 -IP/ICMP Translation Algorithm*. 2011. [Acessado em: 04/11/2016]. Disponível em: <https://tools.ietf.org/html/rfc6145>.

26 DOOLEY, M. *IPv6 Deployment and Management*. 1ª. ed. [S.l.: s.n.], 2013.

27 BAO, C. *RFC 6052 - IPv6 Addressing of IPv4/IPv6 Translators*. 2010. [Acessado em: 26/09/2016]. Disponível em: <https://tools.ietf.org/html/rfc6052>.

28 BAGNULO, M. The nat64/dns64 tool suite for ipv6 transition. v. 1, p. 1–7, 2012.

29 GUHA, E. S. *RFC 5382 - NAT Behavioral Requirements for TCP*. 2008. [Acessado em: 26/09/2016]. Disponível em: <https://tools.ietf.org/html/rfc5382>.

30 CISCO. *NAT64—Stateless versus Stateful*. 2011. [Acessado em: 03/11/2016]. Disponível em: <http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/enterprise-ipv6-solution/white_paper_c11-676277.html>.

31 BAGNULO, M. *DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers*. 2011. [Acessado em: 04/11/2016]. Disponível em: <https://tools.ietf.org/html/rfc6147>.

32 LOTTI, L. P. Sistemas virtualizados – uma visão geral. v. 1, p. 1–8, 2011.

33 ORACLE. *VirtualBox User Manual*. 2014. [Acessado em: 30/11/2016]. Disponível em: <https://www.virtualbox.org/manual/html>.

34 HUBERT, B. *Linux Advanced Routing Traffic Control HOWTO*. 2002. [Acessado em: 04/11/2016]. Disponível em: <http://www.tldp.org/HOWTO/pdf/Adv-Routing-HOWTO.pdf>.

35 ISC. *Berkley Internet Name Daemon (BIND)*. 2016. [Acessado em: 03/11/2016]. Disponível em: <https://www.isc.org/downloads/bind/>.

36 RAHRER, T. Triple-play services quality of experience (qoe) requirements. v. 1, p. 84–86, 2006.