UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ALGORITMO DE SELEÇÃO E FILTRAGEM DE ROTAS PARA REDUÇÃO DAS TABELAS DE ENCAMINHAMENTO

Luiz Fernando Teixeira de Farias

Orientadores
Morganna Carmem Diniz
Sidney Cunha de Lucena

Rio de Janeiro, RJ – BRASIL
SETEMBRO DE 2014

ALGORITMO DE SELEÇÃO E FILTRAGEM DE ROTAS PARA REDUÇÃO DAS TABELAS DE ENCAMINHAMENTO

Luiz Fernando Teixeira de Farias

DISSERTAÇÃO OBTENÇÃO APRESENTADA COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE MESTRE PELO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO (UNIRIO). APROVADA PELA COMISSÃO EXAMINADORA ABAIXO ASSINADA.

Aprovada por:

Morganna Carmem Diniz. D. Sc. - UNIRIO

Sidney Cunha de Lucena. D. Sc. - UNIRIO

Carlos Alberto Vieira Campos. D. Sc. - UNIRIO

Antônio Augusto de Aragão Rocha. D. Sc. - UFF

Ronaldo Moreira Salles. D. Sc. - IME

Rio de Janeiro, RJ – BRASIL

SETEMBRO DE 2014

Farias, Luiz Fernando Teixeira de
F224 Algoritmo de seleção e filtragem de rotas para redução das tabelas de encaminhamento / Luiz Fernando Teixeira de Farias, 2014.
110 f. ; 30 cm

Orientadora: Morgana Carmem Diniz.
Dissertação (Mestrado em Informática) - Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2014.

1. Algoritmos computacionais. 2. Roteadores (Rede de computador). 3. Forwarding Information Base. 4. Routing Information Base. 5. Protocolos de roteamento. I. Diniz, Morgana Carmem. II. Universidade Federal do Estado do Rio de Janeiro. Centro de Ciências Exatas e Tecnológicas. Curso de Mestrado em Informática. III. Título.

CDD – 005.1

Agradecimentos

Agradeço ao Deus do meu coração e da minha compreensão, que iluminou o caminho até o topo montanha, aos meus pais Rumão Tavares de Farias e Maria Jacinta Teixeira de Farias e ao meu filho Tales Florêncio de Farias que alimentaram a determinação e a perseverança para superar os desafios.

Agradeço do fundo do coração aos meus orientadores Prof. Dra. Morganna Carmem Diniz e Prof. Dr. Sidney Cunha de Lucena, pela impecabilidade de agirem como mestres professores. Agradeço também ao professor Carlos Alberto.

Ao Mestre Bruno Fernandes Guedes, um colega que se tornou um amigo. A sua boa índole e caráter demonstraram que as maiores conquistas na vida são as pessoas. E o colega André Martins, que foi companheiro nos momentos de lamentações e nos rodízios de piza.

Também aos meus familiares, amigos, alunos e todos aqueles que contribuíram para a produção deste trabalho.

Resumo

FARIAS, Luiz Fernando Teixeira de. **Algoritmo de seleção e filtragem de rotas para redução das tabelas de encaminhamento**. UNIRIO, 2014. 110 páginas. Dissertação de Mestrado. Departamento de Informática Aplicada, UNIRIO.

O aumento da tabela de roteamento global é um obstáculo para o crescimento da Internet. Além disso, o uso da virtualização em *datacenter* e as Redes Definidas por Software (SDN) intensificaram a demanda por recursos de processamento e armazenamento, aumentando ainda mais os custos das plataformas de roteamento. Este trabalho propõe um algoritmo que reduz o tamanho da tabela de encaminhamento (FIB, *Forwarding Information Base*) a partir de cálculos realizados com base na tabela de roteamento (RIB, *Routing Information Base*) sob a ótica da interface de saída para uma dada rota. Esta abordagem possibilita a redução da FIB de acordo com a topologia da rede empregada. Experimentos com o algoritmo proposto foram realizados com o *software* de roteamento BIRD em máquinas virtuais LXC no *kernel* do Linux. Medidas relativas ao tamanho da tabela de encaminhamento e à latência na rede foram obtidas e analisadas para dois cenários. O primeiro representa dois provedores de serviço de Internet anunciando prefixos com o protocolo BGP. O segundo é um cenário real com todos os prefixos da Internet usando a tabela de roteamento global obtida no servidor *Looking Glass* do projeto Route Views.

Palavras-chaves: Redução da FIB. Protocolos de roteamento. RIB. DFZ.

Abstract

The progressive increase of the global routing table is an obstacle for the Internet growth. Besides, the use of virtualization in datacenter networks and the arise of software-defined networking (SDN) intensified the demand for processing and storage resources, increasing even more the cost of router platforms. This work proposes an algorithm that reduces the forwarding table, or forwarding information base (FIB), from calculations based on the routing table, or routing information base (RIB), from the perspective of the outgoing interface for a given route. This approach allows the reduction of the FIB according to the used network topology. Experiments with the proposed algorithm were done using the BIRD routing software over Linux LXC virtual machines. Measures related to the forwarding table size and network latency were obtained and analysed for a scenario representing two Internet providers announcing network prefixes with the BGP protocol. The second is a real scenario with all prefixes using the Internet global routing table obtained in the Looking Glass server of the Route Views Project.

Key-words: FIB reduction. Routing protocols. RIB. DFZ.

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

API	*Application Programming Interface*
APR	*Aggregation Point Router*
ARP	*Address Resolution Protocol*
AS	*Autonomous System*
ASIC	*Application-Specific Integrated Circuit*
BGP	*Border Gateway Protocol*
CIDR	*Classless Inter-Domain Routing*
DFRT	*Default Free Routing Table*
DFZ	*Default Free Zone*
DUT	*Device Under Test*
ECMP	*Equal cost multi-path*
FIB	*Forwarding Information Base*
FIFA	*Fast incremental FIB aggregation*
IANA	*Internet Assigned Numbers Authority*
ICMP	*Internet Control Message Protocol*
IEEE	*Institute of Electrical and Electronics Engineers*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*

ISP	*Internet Service Provider*
IXP	*Internet Exchange Point*
LPM	*Longest-Prefix Matching*
LXC	*Linux Containers*
MAC	*Media Access Control*
MPLS	*Multi Protocol Label Switching*
MRT	*Multi-Threaded Routing Toolkit*
NDN	*Named Data Networking*
NDP	*Neighbor Discovery Protocol*
ORTC	*Optimal Routing Table Construct*
OSPF	*Open Shortest Path First*
PATRICIA	*Practical Algorithm to Retrieve Information Coded in Alphanumeric*
PTT	*Ponto de troca de tréfego*
RFC	*Request for Comments*
RIB	*Routing Information Base*
RR	*Route-Reflector*
RTT	*Round Trip Time*
SDN	*Software Defined Network*
SMALTA	*Practical and Near-optimal FIB Aggregation*
SRAM	*Static Random Access Memory*
S-VA	*Simple Virtual Aggregation*

TCAM	*Ternary content-addressable memory*
VETH	*Virtual Ethernet Device*
VLSM	*Variable Lenght Subnet Mask*
VPN	*Virtual Private Network*
VRF	*Virtual Routing and Forwarding*

Sumário

1 Introdução

1.1 Considerações Iniciais

Protocolos de roteamento como o BGP (*Border Gateway Protocol*) ou OSPF (*Open Shortest Path First*) costumam ser usados pelos roteadores dos provedores de serviço para descobrir o caminho a ser percorrido por cada pacote até seu respectivo destino. Os resultados são armazenados numa tabela denominada base de informação de roteamento (RIB, *Routing Information Base*). Nesta tabela são registrados todos os possíveis caminhos até os prefixos de rede aprendidos por todos os protocolos de roteamento que estão em operação. Na base de informação de encaminhamento (FIB, *Forwarding Information Base*) estão registrados os prefixos da rede de destino e o endereço IP do próximo roteador (*next hop*), já sanados quaisquer conflitos, ou seja, caso existam diferentes rotas para o mesmo destino. A diferença entre o conteúdo da RIB e da FIB está na simplicidade necessária para se obter um melhor desempenho no envio das informações. A RIB possui o registro de todos os detalhes das rotas, enquanto a FIB possui o necessário para encaminhar o pacote até o próximo salto.

Para criar um quadro que levará o pacote até a próxima interface (*framming*) é necessário que o endereço físico (HW-ADDR, *hardware address*) do próximo roteador seja conhecido. Cada tipo de endereço de alto nível tem um protocolo específico para fazer a associação com os endereços dos dispositivos físicos. Por exemplo, o endereço IP versão 4 é traduzido com o protocolo de tradução de endereço (ARP, *Address Resolution Protocol*), enquanto o endereço IP versão 6 é traduzido com outro protocolo denominado "protocolo de descoberta da vizinhança" (NDP, *Neighbor Discovery Protocol*) (NARTEN et al., 2007). O resultado é armazenado na tabela de endereçamento de *hardware* (*cache* ARP) (PLUMMER, 1982). Os quadros que chegam na interface de entrada (RX) são

processados e enviados para uma dada saída (TX) de acordo com o *cache* das tabelas FIB e ARP na memória do dispositivo.

Os elementos que controlam a rede formam o plano de controle, enquanto os elementos responsáveis por fazer o encaminhamento dos pacotes pertencem ao plano de dados (GUEDES et al., 2012). O protocolo de roteamento e a RIB são elementos do plano de controle, enquanto a FIB e a tabela ARP são elementos do plano de dados (MOHAMED et al., 2012). A ligação entre esses elementos é observada na Figura 1.

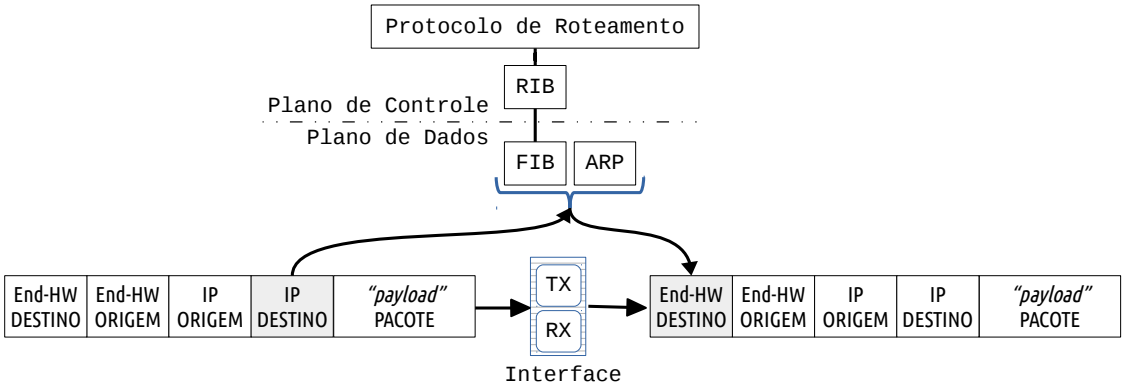


Figura 1 – Separação de elementos usados no encaminhamento dos pacotes.

A virtualização de redes proporciona o compartilhamento da capacidade dos componentes de uma rede física, tornando possível estabelecer múltiplas infraestruturas lógicas distintas sobre os mesmos componentes (ARAÚJO et al., 2012). Novas arquiteturas de rede, com foco de uso em *Datacenters* e *Cloud Computing*, impulsionam a adoção de técnicas de virtualização (ROTHENBERG et al., 2012) por oferecerem flexibilidade para realocação de recursos físicos.

Uma topologia de rede com vários roteadores pode ser obtida em um único computador através de virtualização. A virtualização possibilita o compartilhamento dos recursos computacionais, incluindo o plano de dados, entre os diversos elementos virtuais. No ambiente virtual em que suítes de roteamento (SRs) são executadas em diferentes máquinas virtuais (MVs), as configurações de rede, as RIBs e a FIBs são isoladas e independentes entre si. Por exemplo, uma topologia formada com cem roteadores foi

criada sobre um único *kernel* Linux nos experimentos executados em (FARIAS et al., 2013). Neste cenário, o enlace entre as SRs foi obtido com uma ponte Linux e as FIBs que compartilham a mesma memória física. No caso de ser estabelecida uma conectividade com uma rede física, a memória do dispositivo teria que ser suficientemente grande para acomodar as tabelas de encaminhamento resultantes das rotas aprendidas desta rede física.

Uma situação parecida ocorre em *datacenters* quando o recurso da virtualização de redes é usado. Um estudo sobre demandas de *hardware* que suportem o aumento da FIB com as exigências da virtualização é apresentado em (ROTTENSTREICH et al., 2013). Problema similar seria observado nas redes definidas por *software* com o uso da *sobreposição de redes*. Nesta técnica, um dispositivo pode ser programado de maneira a participar de diferentes topologias simultaneamente, aumentando o espaço de memória necessário para o armazenamento das regras de fluxo referentes às respectivas FIBs (ARAÚJO et al., 2012).

O aumento do tamanho das FIBs é um obstáculo para o crescimento do número de utilizadores da Internet (MEYER et al., 2007) e a necessidade de soluções torna-se prioritária (SÁNCHEZ, 2013). A consulta do próximo salto aumenta o tempo para os roteadores realizarem o envio dos dados, enquanto o tempo para atualização dos prefixos influencia a convergência após falhas, ou seja, a atualização da FIB após modificações na RIB (MOHAMED et al., 2012). Em (BONAVENTURE et al., 2007) foi calculado que 40 a 55% do tempo total de convergência em provedores de serviços (ISP, *Internet Service Providers*) é consumido na atualização da FIB, e seu tamanho influencia diretamente neste tempo de atualização. O tamanho das FIBs e sua influência na eficiência das redes privadas virtuais (VPN - *Virtual Private Network*) de larga escala é observado por Mai e Du (2013).

O aumento da tabela de roteamento global é causado pela forma como os prefixos de rede são administrados (KUMAR; KUMAR, 2013) e pela demanda por novos endereços IP. Esta situação é mais evidente na região sem rota padrão (DFZ, *Default Free Zone*) empregada no núcleo da Internet (BERKOWITZ et al., 2005). Roteadores usados nos ISPs

nível 1 (*TIER-1*), por exemplo, não configuram a rota padrão (0.0.0.0/0 no IPv4 e ::/0 no IPv6), ou seja, as RIBs nestes roteadores possuem rotas definidas para todos os prefixos de rede da Internet. Estas tabelas são denominadas DFRT (*Default Free Routing Table*) e, atualmente, possuem cerca de 500 mil prefixos (HUSTON, 2013).

1.2 Motivação

O gráfico na Figura 2 ilustra o aumento do número de prefixos anunciados na Internet usando o BGP. Este crescimento demanda a utilização de *hardware* que seja capaz de armazenar e processar os dados melhorando os tempos de resposta. Isto resulta em equipamentos de altíssimo custo com *chips* de memórias de alta velocidade (SRAM, *Static Random Access Memory*) criados para organizar a tabela de encaminhamento e obter otimização nas buscas dos dados armazenados (TCAM, *Ternary content-addressable memory*) e processadores customizados para este propósito específico (ASIC, *Application-Specific Integrated Circuit*) (CISCO Systems, 2012).

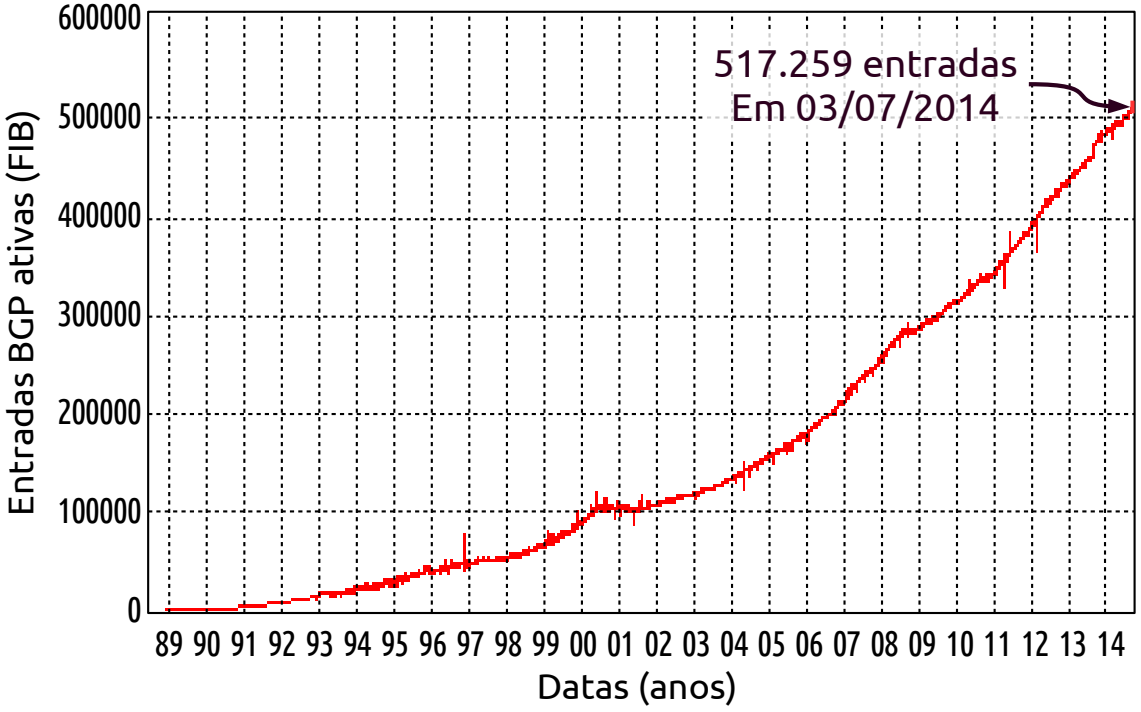


Figura 2 – Crescimento da tabela de roteamento global.

Fonte: <http://bgp.potaroo.net>

O problema existente no sistema de endereçamento e roteamento adotado na Internet é evidenciado com o aumento do número de usuários. Da mesma forma, fatores relacionados à administração dos endereços IP contribuem para o aumento das tabelas de roteamento (KUMAR; KUMAR, 2013).

A discussão sobre o problema proporcionou hipóteses muito diferentes. Por exemplo, o trabalho apresentado por Rottenstreich et al. (2013) descreve o problema como urgente, apresentando uma expectativa para *chips* de memórias que ainda estão em fase de concepção, e já são necessários para acomodar a demanda do aumento das entradas na FIB em vários milhões. Em contrapartida, Fall et al. (2009) apresenta em sua pesquisa um ponto de vista diferente, demonstrando com a *Lei de Moore* (MOORE, 1965) que as inovações tecnológicas resolverão os problemas conforme forem surgindo.

1.3 Objetivos

Essa alteração no conteúdo da FIB deve reduzir o tempo para encaminhar os pacotes mantendo a alcançabilidade para todos os destinos.

Assim sendo, propõe-se aqui um algoritmo que filtra grupos de prefixos de rede selecionando registros na tabela de roteamento. Esta remoção de um conjunto de entradas na FIB não altera o mecanismo de encaminhamento se os prefixos informados pela RIB forem mesclados de acordo com o enlace de saída. Dessa forma, a utilização de um único registro que englobe grupos de endereços na FIB não interfere no encaminhamento do pacote na direção correta. A ideia tem como base a observação de que o próximo salto representa o grupo de prefixos de rede que são alcançados naquela direção. No algoritmo proposto, os prefixos informados pela RIB são separados de acordo com o endereço de próximo salto e depois gravados na FIB.

Esta proposta contribui com uma solução de redução da FIB sob a ótica da saída da interface da rede. Dessa forma, além de se adequar com diferentes tipos de

endereçamento lógico (por exemplo, o IPv6), possibilita a utilização simultânea de outros mecanismos que também são usados para reduzir o tamanho da FIB, como o S-VA (*Simple Virtual Aggregation*) (BALLANI et al., 2009) e o CIDR (*Classless Inter-Domain Routing*) (FULLER; LI, 2006).

Para validar o funcionamento do sistema proposto em ambientes de maior escala, a tabela de roteamento global, obtida na base de dados do projeto Route Views (MEYER, 2014), foi filtrada e agregada com o algoritmo. Os resultados obtidos mostram uma redução de 72,13% no tamanho da FIB, considerando um caso particular representativo de interligação entre provedores de Internet.

1.4 Estrutura da Dissertação

A dissertação está organizada da seguinte forma. No Capítulo 2 são apresentadas as pesquisas anteriores associadas ao problema do aumento da tabela de roteamento global com a descrição das principais técnicas usadas para agregação e filtragem. Também é explicado os fundamentos do endereçamento usado na Internet, a hierarquia de ligação entre os sistemas autônomos e a estrutura de dados das tabelas empregadas para o encaminhamento dos pacotes. No Capítulo 3 os objetivos e a estrutura detalhada dos módulos conceituais são descritos juntamente com a implementação do algoritmo proposto para comprovar a hipótese. Também há a descrição da complexidade, a comparação com o FIFA (LIU et al., 2013b) e o SMALTA (UZMI et al., 2011) e considerações para integração com outras plataformas. Em seguida, o Capítulo 4 explica a forma como a proposta foi avaliada com métricas obtidas num cenário criado em ambiente virtual que simula aquele usado na região da Internet com roteadores sem a rota padrão (DFZ *Default Free Zone*), o procedimento de monitoramento e a interpretação dos resultados. A avaliação da proposta no cenário real é feito no Capítulo 5. Para isso, é explicado como a base de dados do projeto Route Views foi acessada para obtenção da RIB usada na Internet e a análise da FIB reduzida, obtida com a aplicação do algoritmo proposto. Ao final, o Capítulo 6 discute as contribuições

que a pesquisa acrescenta ao tema e as publicações decorrentes. Além disso, as vantagens e as limitações do modelo proposto são discutidas, assim como sugestões para trabalhos futuros relacionados com este tema.

2 Pesquisas Relacionadas

Nas pesquisas realizadas para redução do tamanho da tabela de encaminhamento (FIB), basicamente duas técnicas são usadas: agregação e filtragem. A agregação é uma técnica que mescla os prefixos de destino que compartilham os mesmos caminhos usando um algoritmo implementado internamente. Esta solução tem menor custo, por não exigir mudanças de *hardware*, apresentar maior simplicidade do que as soluções que exigem mudanças na topologia e não afetar o comportamento externo do roteador pois não requer alterações nos protocolos. Estas características são usadas como justificativas nas propostas usadas como referência neste trabalho.

A filtragem usa a ideia de selecionar na base de informação de roteamento apenas os prefixos que são necessários para realizar o encaminhamento dos pacotes. Os algoritmos propostos na literatura utilizam características da topologia ou do *hardware* para selecionar os registros que serão gravados na FIB, dentre aqueles que foram obtidos na RIB. Por exemplo, a tecnologia de encaminhamento e roteamento em redes privadas virtuais (VRF, *Virtual Routing and Forwarding*) foi criada para manter o desempenho nos roteadores quando provedores de serviços de Internet fornecem para os seus clientes serviços de redes privadas virtuais (VPN) (ROSEN; REKHTER, 2006). Esta tecnologia mantém uma tabela de roteamento e encaminhamento para cada conexão VPN criada nos roteadores de borda, que dão acesso ao núcleo da rede.

Entre as tecnologias de agregação e filtragem existentes, as seguintes técnicas serão discutidas ao longo deste capítulo:

Sumarização de rotas

O cálculo do maior prefixo comum possibilita que a tecnologia usada no roteamento entre domínios (CIDR, *Classless Inter-Domain Routing*) (FULLER; LI, 2006) trans-

mita anúncios com os prefixos agregados, reduzindo a tabela de roteamento dos roteadores vizinhos.

Agregação com algoritmos de busca em árvores

Os algoritmos *Optimal Routing Table Construct* (DRAVES et al., 1999), *Practical and Near-optimal FIB Aggregation* (UZMI et al., 2011) e *Fast incremental FIB aggregation* (LIU et al., 2013b) utilizam estruturas de árvores binárias para agregar os prefixos obtendo a redução da FIB e, consequentemente, a melhora no desempenho no mecanismo de encaminhamento dos roteadores.

Simple Virtual Aggregation

O S-VA (RASZUK et al., 2012) é uma opção que os provedores de serviços têm para reduzir a tabela de encaminhamento usando algoritmos de agregação junto com o protocolo MPLS (*Multiprotocol Label Switching*) (ROSEN et al., 2001).

Protocol based selective FIB download

O trabalho em (MOHAMED et al., 2012) propõe um algoritmo de filtragem com base na técnica de encaminhamento e roteamento em redes privadas virtuais (VRF, *Virtual Routing and Forwarding*) para reduzir a tabela de encaminhamento em roteadores com tecnologia distribuída.

Além destes trabalhos, as redes definidas por software (SDN, *Software Defined Network*) têm recebido propostas para solucionar os problemas associados com a demanda de processamento dos dispositivos que operam nestas redes. As limitações dos componentes de *hardware* empregados nas redes tradicionais também são observadas nesta arquitetura, por isso as soluções implementadas podem trazer benefícios para provedores de serviços e outros usuários de SDN. Além disso, a possibilidade da redução dos recursos em roteadores que trabalham em conjunto com controladores *OpenFlow* (MCKEOWN et al., 2008) em redes híbridas fomenta a pesquisa acadêmica (SHAHZAD, 2013).

Neste Capítulo, também é analisada a implementação da tecnologia de agregação virtual (S-VA) em dispositivos *OpenFlow*. No trabalho de Sánchez (2013) são feitas sugestões que permitem automatizar a agregação das rotas em SDN.

2.1 Sumarização de Rotas

Os endereços IP versão 4 são formados por quatro octetos que representam a identificação da rede e a identificação do *host* nesta rede. Inicialmente, os endereços *classfull* determinavam a quantidade de participantes para um número fixo de redes, porém a demanda por novas redes resultou na utilização de máscaras para identificar grupos formados a partir da divisão das redes *classfull* (COMER, 2006). O prefixo é o primeiro endereço IP de um bloco usado para identificar a rede.

2.1.1 VLSM (*Variable Lenght Subnet Mask*)

O uso de máscaras para sub-redes com tamanho variável (VLSM, *Variable Lenght Subnet Mask*) permite agrupamentos menores, que se encaixem nas diferentes requisições por blocos de números IPs . A característica deste mecanismo é que ele foi criado para ser empregado na divisão de uma rede em grupos menores. Por exemplo, na Tabela 1 cada linha identifica o primeiro e último IP de uma rede com a quantidade de endereços IPs definida pela máscara de rede. Cada nova rede (linha) começa com o IP seguinte ao assinalado na linha anterior e um número menor associado com a máscara.

IP inicial - Final	**Máscara**	**IP e Máscara em binário**
192.168.1.0	/27	11000000.10101000.0000001.00000000
192.168.1.31	*(32 IPs)*	11000000.10101000.0000001.11100000
192.168.1.32	/29	11000000.10101000.0000001.00100000
192.168.1.39	*(8 IPs)*	11000000.10101000.0000001.11111000
192.168.1.40	/30	11000000.10101000.0000001.00101000
192.168.1.43	*(4 IPs)*	11000000.10101000.0000001.11111100

Tabela 1 – Exemplo de atribuição de prefixos.

2.1.2 LPM (*Longest-Prefix Matching*)

Um forma de agregar os prefixos é através do cálculo do maior prefixo comum (LPM, *Longest-Prefix Matching*). O diagrama ilustrado na Figura 3 representa alguns blocos obtidos com a manipulação da máscara de rede a partir do prefixo de rede 192.168.1.0/25. O prefixo inicial é dividido consecutivamente até resultar nos seis prefixos finais usados neste exemplo. Por isso, o cálculo da máscara de rede em comum para todos os prefixos resulta na mesclagem dos endereços IPs em um prefixo formado pelos bits que são comuns (leitura da esquerda para direita da Figura 3).

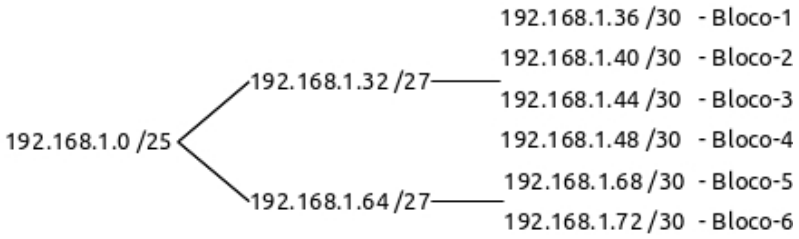


Figura 3 – Blocos de endereços IP.

Na Tabela 2, os quatro primeiros prefixos (grupo 1) têm o último octeto convertido para binário de maneira a calcular o maior prefixo em comum. A máscara é obtida com a posição do último bit em comum. Todos os prefixos do primeiro grupo têm o valor binário 001* no último octeto, ou seja, possuí o prefixo 001 e todas as combinações possíveis nos bits do sufixo (*). Enquanto no segundo grupo tem o prefixo 0100*.

2.1.3 CIDR (*Classless Inter-Domain Routing*)

No modelo hierárquico de endereçamento recomendado para ser usado na Internet, cada provedor de serviço (ISP) recebe um bloco de endereços IP de maneira que pode fazer a atribuição para os clientes (KLEINROCK; KAMOUN, 1977). Por exemplo, se o ISP receber o prefixo `200.20.0.0/16` ele poderá decidir fornecer para dezesseis outros provedores de serviços blocos com máscara /20, a partir do bloco 200.20.0.0/20 até o bloco 200.20.240.0/20, ou duzentos e cinquenta e seis endereços para duzentos e cinquenta e seis redes usando a máscara /24, a partir do bloco 200.20.0.0/24 até o bloco 200.20.255.0/24.

Prefixo	**Octeto-4 em binário**	**Agregado**
192.168.1.36/30	192.168.1.**001**00100	
192.168.1.40/30	192.168.1.**001**01000	
192.168.1.44/30	192.168.1.**001**01100	
192.168.1.48/30	192.168.1.**001**10000	
Prefixo comum:	192.168.1.00100000	192.168.1.32/27
192.168.1.68/30	192.168.1.**0100**0100	
192.168.1.72/30	192.168.1.**0100**1000	
Prefixo comum:	192.168.1.01000000	192.168.1.64/28
192.168.1.32/27	192.168.1.**0**0110000	
192.168.1.64/28	192.168.1.**0**1000000	
Prefixo comum:	192.168.1.00000000	192.168.1.0/25

Tabela 2 – Exemplo de cálculo do *Longest-Prefix Matching*

Dessa forma, a tecnologia de endereçamento entre domínios (CIDR, *Classless Inter-Domain Routing*) pode ser empregado (FULLER; LI, 2006) .

Outra vantagem do uso desta tecnologia é a configuração nos equipamentos de rede dos anúncios de prefixos após calcular o *Longest-Prefix Matching*. O resultado é a redução da tabela de roteamento nos equipamentos vizinhos, pois apenas o prefixo resultante é usado para determinar a melhor rota.

2.2 Agregação com algoritmos de busca em árvores

A base de encaminhamento das informações (FIB) é uma tabela formada pelo campo *prefixo da rede de destino*, que é usado para indexar os registros, o campo *próximo salto*, que registra o endereço lógico (número IP) do próximo roteador do caminho, e o campo *interface*, que identifica a saída usada para enviar o pacote (TROTTER, 2001).

A agregação da FIB pode ser obtida com mecanismos que organizam os prefixos em estruturas de árvores binárias. Esta representação dos dados permite encontrar o próximo salto e fazer alterações nos valores da tabela mais rapidamente (COMER, 2006).

O construtor da tabela de roteamento ideal (ORTC), que utiliza o algoritmo de

busca em árvores (TRIE) para obter a redução da tabela de encaminhamento, é usado como referência na proposta do SMALTA e do FIFA, que utiliza o algoritmo de busca rápida em árvores (PATRICIA).

2.2.1 Árvore Binária

Algoritmos de busca em árvores de prefixos (TRIE, *reTRIEve prefix tree*) podem ser usados em estruturas com dados de tamanho variável, como os prefixos de destino na FIB (DRAVES et al., 1999). A ideia é usar os bits do prefixo para indicar o caminho usado na ramificação. Quando o bit é zero, a ramificação é feita para esquerda, e quando o bit é 1, a ramificação é para direita. A vantagem é a simplicidade da representação e as desvantagens são a altura que o endereço IP faz com que a árvore possua (32 nós) e os trechos que são comuns aos prefixos. Na Tabela 3 estão representados em binário o último octeto dos prefixos usados na construção da árvore binária usada na exemplificação.

Prefixo	**Representação em binário**	
192.168.1.36/30	192.168.1.	00100100
192.168.1.40/30	192.168.1.	00101000
192.168.1.44/30	192.168.1.	00101100
192.168.1.48/30	192.168.1.	00110000
192.168.1.68/30	192.168.1.	01000100
192.168.1.72/30	192.168.1.	01001000

Tabela 3 – Conjunto de prefixos.

Após a ordenação dos dados, cada bit do prefixo é usado para determinar o caminho. Os nós interiores da árvore representam dois ou mais prefixos e cada nó exterior é um prefixo exclusivo. A representação da árvore binária a partir da raiz comum de todos os prefixos é ilustrada na (Figura 4). Neste desenho, o número de bits usados para representar os nós dependem dos conjuntos formados. Por exemplo, o primeiro conjunto é formado com um único *bit* e representa a raiz da árvore. Para o próximo desagrupamento, um único bit é necessário e por isso a máscara sofre um incremento da unidade. O final da ramificação representa o prefixo que será agregado.

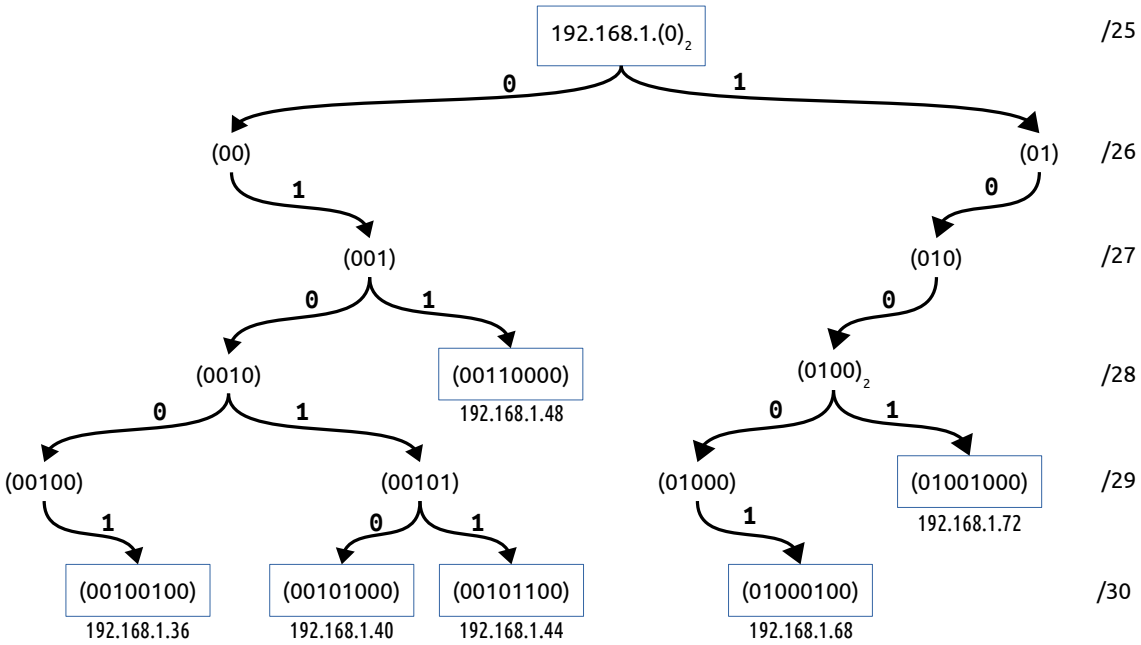


Figura 4 – Representação da Árvore binária.

O prefixo agregado é obtido nos nós comuns das ramificações. Por exemplo, o endereço 192.168.1.0(001*) com máscara /27 agrega os prefixos de rede 192.168.1.36, 192.168.1.40 , 192.168.1.44 e 192.168.1.48, enquanto que o endereço 192.168.1.0(0100) com máscara /28 agrega os prefixos 192.168.1.68 e 192.168.1.72. Este é o mesmo resultado que foi apresentado anteriormente na Tabela 2.

2.2.2 PATRICIA (*Practical Algorithm to Retrieve Information Coded in Alphanumeric*)

O algoritmo *PATRICIA* foi criado para reduzir a árvore binária, obtendo uma otimização na busca (MORRISON, 1968). A desvantagem deste método é o aumento da carga de processamento para a criação da árvore. No entanto, a existência de um maior número de buscas em relação ao quantidade de vezes que à tabela é alterada justifica a utilização do método.

A ideia é usar o prefixo como chave, permitindo que buscas sejam realizadas na estrutura com partes da chave. Por exemplo, se uma árvore tiver apenas dois pre-

fixos, `192.168.1.0/24` e `192.168.2.0/24`, o primeiro encontro na busca seria `128.0.0.0` (primeiro bit 1) e o segundo resultaria no término. Caso fosse uma árvore binária, cada um dos bits teria que ser procurado, resultando em vinte e quatro encontros positivos, pois estes são os bits em comum entre os dois prefixos.

O mecanismo funciona com a utilização de algum critério pré-definido para determinar o número do saltos. A chave é determinada pela posição na árvore, ao contrário da árvore binária na qual a chave é associada com o nó. Todos os nós tem que conter uma chave que identifica o último bit do prefixo comum, e os saltos representam o bit que é diferente.

No exemplo ilustrado na Figura 5, os primeiros bits mais significativos são comuns para os dois octeto, por isso só há uma ramificação. O primeiro bit em comum é usado para representar a ligação até o próximo bit que é diferente na posição vinte e nove, que é indicada no nó. As ramificações terminam nos prefixos exclusivos.

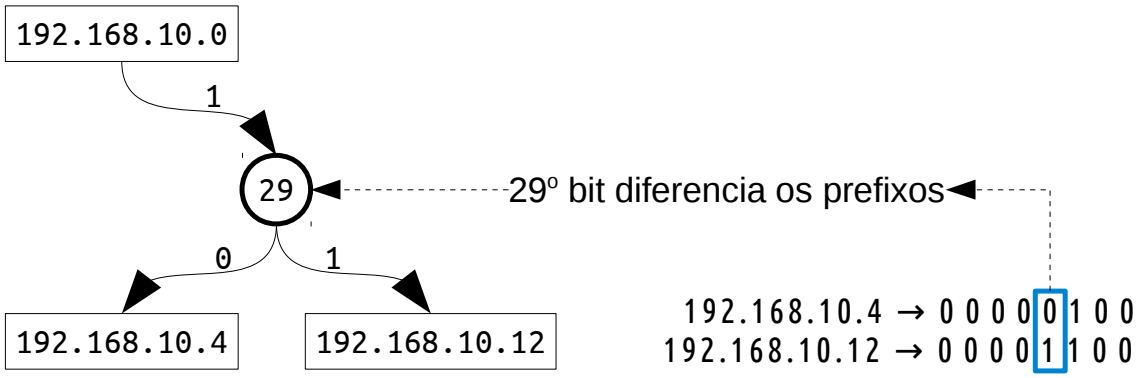


Figura 5 – Exemplo da árvore *PATRICIA*.

2.2.3 ORTC (*Optimal Routing Table Constructor*)

O construtor da tabela de roteamento ideal (ORTC, *Optimal Routing Table Constructor*) é uma referência nas pesquisas que utilizam a agregação dos prefixos para obter a redução da tabela de encaminhamento. De acordo com Draves et al. (1999), este mecanismo diminuiu aproximadamente 40% o número de entradas na tabela de encaminhamento (FIB) obtida

nos roteadores usados em grandes provedores de serviços. Esta solução tem como premissa que o cálculo das rotas obtidas com o protocolo BGP resulta em caminhos *não ideais* devido ao grande número de anúncios recebidos.

O algoritmo é aplicável em tabelas que possuem correspondência entre os prefixos (*Longest-Prefix Matching*) IP versão 4. O resultado é obtido localmente em cada roteador com a divisão e o agrupamento de blocos de endereços que possuem o mesmo *próximo salto* e formam um menor número de entradas. Por exemplo, o bloco 000* pode ser dividido formando dois novos blocos 0000* e 0001*, ou pode ser agrupado no bloco 00*. Estas operações são opções para preservar a múltipla informação do próximo salto ou obter melhor taxas de redução.

A representação da tabela de rotas na árvore binária utiliza indicadores do endereço do próximo salto nos nós. O nível superior é o caminho usado pelo nó inferior. Por isso, na raiz é colocado o indicador da rota padrão, de maneira a garantir que cada nó filho tenha um próximo salto. Draves et al. (1999) sugere empregar o zero como uma imitação do caminho (*dummy next hop*) para o caso do roteador que não tem a rota padrão configurada, como os usados no núcleo da Internet (DFZ, *Default Free Zone*). Esta rota deve ser desconsiderada no cálculo do prefixo correspondente. O identificador do salto é colocado na "última folha", e em cada ramo é colocado o *bit* (0 ou 1) do prefixo específico daquela rota.

As rotas listadas na Tabela 4 são utilizadas para criar a representação da árvore binária (Figura 6). Nesta ilustração, a raiz é zero (sem rota padrão) e as seis rotas (próximos saltos) são indicadas nas últimas folhas da árvore. Os ramos que ligam a raiz até a primeira rota têm como identificadores os *bits* $0 \rightarrow 0 \rightarrow 1 \rightarrow 0 \rightarrow 0 \rightarrow 1$ associados ao prefixo. Os dois últimos bits não são representados porque são comuns para todos os prefixos com a mesma rota.

A agregação é o resultado obtido com a execução de três passos do algoritmo.

Prefixo	**Próximo Salto**
00100100	1
00101000	1
00101100	1
00110000	1
01000100	2
01001000	2

Tabela 4 – Rotas usadas na árvore binária.

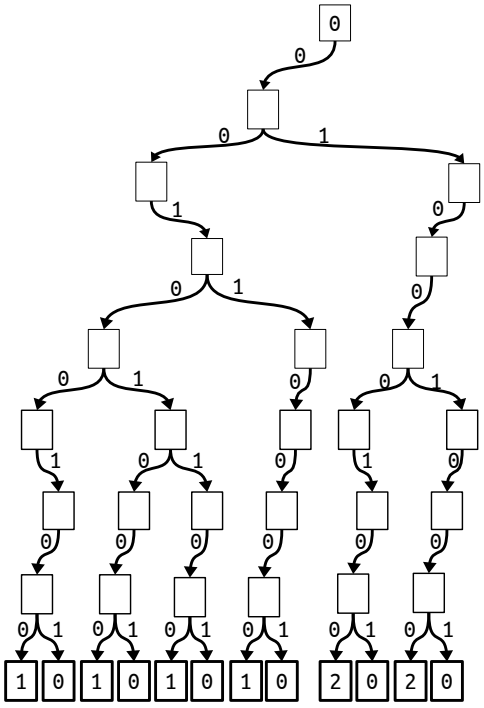


Figura 6 – Árvore binária.

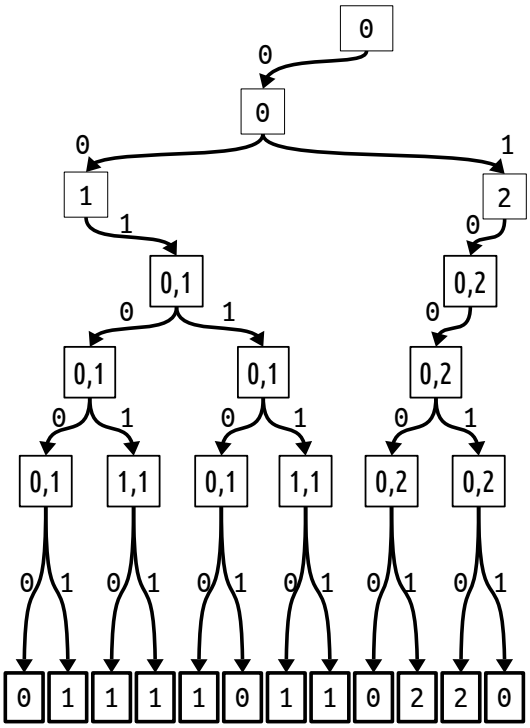


Figura 7 – ORTC: Árvore normalizada.

O primeiro passo é a normalização da tabela.

Na representação em árvore binária cada nó é analisado para determinar a quantidade de ramificações. Um novo nó é atribuído quando só existe uma folha, de maneira que em cada ramificação exista nenhuma ou duas folhas. Esta operação começa no nível hierárquico superior (raiz) e resulta na ampliação da árvore. A ilustração da Figura 7 representa esta sequencia.

No segundo, são calculados os possíveis caminhos redundantes.

A partir do nível hierárquico inferior cada nó é preenchido com os valores dos

nós inferiores, usando uma vírgula para diferenciar as duas possíveis rotas. Os nós intermediários que não possuem valores nem ramificações, que foram criados quando a árvore foi normalizada, recebem o valor nulo da raiz. A raiz não sofre ação desta operação, permanecendo inalterada. Estas indicações foram omitidas na representação da Figura 7 de maneira a simplificar o exemplo.

O terceiro passo é a atribuição do *próximo salto*.

A partir da raiz, cada ramificação é analisada para identificar o valor comum em todos os níveis hierárquicos. O valor encontrado é atribuído ao nó e as ramificações inferiores são removidas. No caso de não ser encontrado um valor exclusivo, a atribuição é removida (valor *null*) e o próximo nível inferior é analisado. A árvore binária resultante é ilustrada na Figura 8.

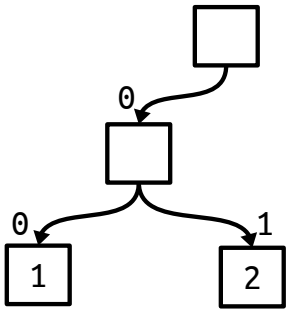


Figura 8 – Arvore binária binária resultante do algoritmo ORTC.

Nos nós estão os valores dos *próximos saltos* e as ramificações são diferenciadas com os bits que formam o prefixo de destino. Por exemplo, os prefixos da Tabela 3 são diferenciados pelos dois primeiros bits associados com as ramificações da Figura 8

2.2.4 SMALTA (*Saving Memory and Lookup Time via Aggregation*)

O SMALTA é um mecanismo que utiliza a proposta do ORTC (Subseção 2.2.3) para obter simultaneamente a redução do tamanho e a redução do tempo de busca na tabela de encaminhamento (FIB *lookup*). A implementação foi demonstrada com a suíte de roteamento Quagga e as métricas calculadas com dados coletados com o protocolo BGP

em um provedor de nível 1 (*Tier-1*) armazenados na base de dados do projeto *Route Views* (MEYER, 2014).

Duas tabelas são usadas para armazenar os resultados marcados na RIB como a melhor rota (UZMI et al., 2011).

- A estrutura de dados "Árvore Original" (OT, *Original Tree*) é usada para calcular os prefixos agregados usando as fases de normalização, cálculo dos possíveis caminhos e atribuição do *próximo salto* do ORTC (Subseção 2.2.3).
- A tabela de encaminhamento (FIB) é configurada com os dados obtidos na "Árvore Agregada" (AT, *Aggregate Tree*), que é uma segunda estrutura usada para armazenar os resultados da *OT* e as atualizações de rotas. Essa segunda tabela evita que todos os prefixos sejam recalculados, resultando em menor tempo para atualizar os dados na FIB .

2.2.5 FIFA (*Fast Incremental FIB Aggregation*)

A proposta de Liu et al. (2013b) é uma melhoria do construtor de tabela de roteamento ideal (ORTC), pois permite a atualização gradual de um prefixo qualquer sem a necessidade de refazer toda a estrutura em árvore na FIB. Como descrito na Subseção 2.2.2, a construção de uma árvore que possibilite maior eficiência na busca demanda maior carga de processamento. Por isso, o FIFA foi concebido com três algoritmos distintos que podem ser implementados de acordo com a demanda do operador.

FIFA-S É o algoritmo que faz a agregação das partes da FIB alteradas buscando a melhor taxa de redução. Esta opção obtém maiores taxas de agregação porque todas as mudanças são tratadas, porém com um alto custo computacional das repetidas execuções.

FIFA-T Este é o algoritmo usado para reduzir a carga de processamento. Realiza a re-agregação a partir da raiz quando as alterações alcançam um limite determinado.

FIFA-H É um esquema híbrido que busca um equilíbrio entre uma melhor taxa de agregação e o intervalo entre as alterações. Este algoritmo é mais rápido do que o FIFA-T, pois a quantidade de prefixos que têm que ser alterados é menor e exige menos do processador porque o intervalo entre as execuções é maior do que o FIFA-S.

2.3 *Simple Virtual Aggregation* (S-VA)

A ideia chave é a divisão da tabela de roteamento em prefixos virtuais que são distribuídos entre roteadores participantes de acordo com a organização da rede virtual criada pelo ViAggre (SVA, *Simple Virtual Aggregation*). A técnica consiste na redução da tabela de encaminhamento nos equipamentos que estão interconectados por túneis virtuais, criados com o protocolo MPLS (*Multi Protocol Label Switching*). Os provedores de serviço (ISP) podem configurar os roteadores para manter a RIB completa enquanto os prefixos armazenados na FIB são distribuídos em uma topologia virtual.

O roteador ponto de agregação (APR, *Aggregation Point Router*) possui na FIB apenas uma fração dos prefixos assinalados na RIB. O calculo do maior prefixo em comum (*Longest-Prefix Matching*) desse agregado é anunciado como o prefixo virtual (VP, *virtual prefix*) para os demais roteadores (Tabela 5). Dessa forma, os outros equipamentos mantém na FIB apenas o VP e a identificação do túnel usado para alcançar o APR designado, que por sua vez possuem os endereços dos próximos saltos e os túneis que deverão ser usados pelo pacotes até o destino.

Por exemplo, na Figura 9 os dois roteadores APR1 e APR2 são configurados como ponto de de agregação (APR, *Aggregation Point Router*). Quando um pacote entra na borda do AS1 ele é encapsulado e comutado pelo roteador participante até o APR que armazena na FIB do agregado.

Destino	Representação em binário	*Virtual Prefix*
197.35.16.0/24	197.35.(00001000.00000000)	197.35.16.0/20
197.35.20.0/24	197.35.(00001100.00000000)	197.35.16.0/20
197.35.30.0/24	197.35.(00001110.00000000)	197.35.16.0/20
202.16.80.128/24	202.16.(01010000.10000000)	202.16.64.0/18
202.16.84.192/24	202.16.(01010100.11000000)	202.16.64.0/18
202.16.126.128/24	202.16.(01111110.10000000)s	202.16.64.0/18

Tabela 5 – Endereços separados em prefixos virtuais.

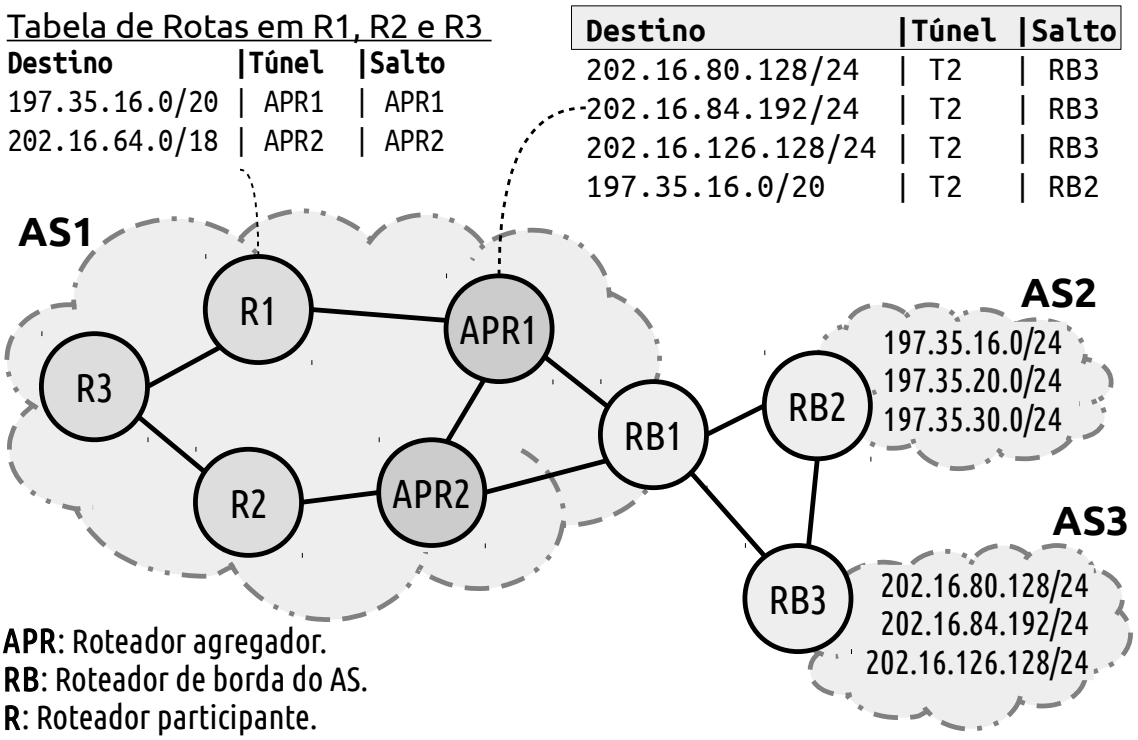


Figura 9 – S-VA: *Simple Virtual Aggregation*.

2.4 Filtragem e Distribuição da FIB

Os prefixos usados com maior frequência geralmente são copiados da memória principal do roteador para a interface conectada no enlace, de maneira obter uma melhora no desempenho do mecanismo de encaminhamento dos pacotes (TROTTER, 2001).

Quando um pacote chega na entrada física, a leitura na *cache da FIB* fornece os endereços associados com o próximo salto e a interface de saída que deverão ser usadas. Nos casos em que o prefixo de destino não é encontrado (*cache miss*), o pacote é

encaminhado para a memória principal do roteador onde é armazenada a FIB completa. Este mecanismo influencia diretamente no desempenho do roteador, pois o processador é consultado apenas quando o pacote não é encontrado na FIB da interface, o que reduz a carga de processamento deixando recursos livres para outras operações.

2.4.1 Arquitetura FIB Centralizada e Distribuída

Os recursos computacionais são usados em rotinas executadas no plano de dados, como a manutenção da tabela de encaminhamento, nas rotinas do plano de controle, como a execução dos protocolos de roteamento (por exemplo o BGP) e a manutenção da base de informação de roteamento (RIB), e no plano de gerenciamento usado para interação com o usuário, através de interfaces de linhas de comando (*shell* CLI) ou com protocolos de acesso remoto (por exemplo o SSH). Os roteadores tradicionais, chamados roteadores centralizados, possuem na placa principal elementos pertencentes a todos os planos, como a RIB e a FIB, e nas interfaces estão apenas elementos do plano de dados, como a *cache FIB* (Figura 10).

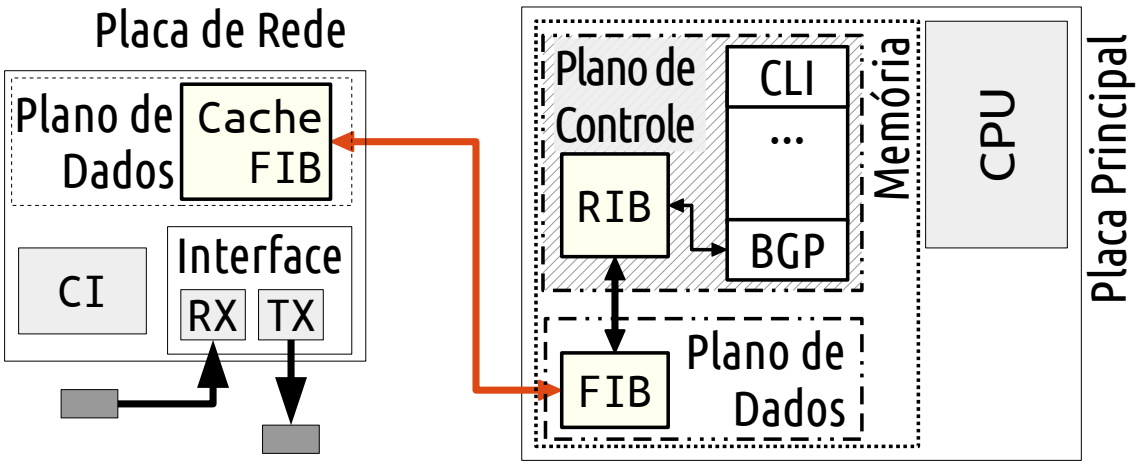


Figura 10 – Arquitetura FIB centralizada.

O aumento da tabela de roteamento global torna a *cache* um obstáculo para o acesso da memória principal em algumas situações, como nos roteadores usados em provedores de acesso do núcleo da Internet. Isso ocorre porque a maior quantidade de

prefixos resulta na releitura da memória principal para atualizar a *cache* com as rotas necessárias. O resultado é a redução da eficiência no encaminhamento dos pacotes na mesma proporção de crescimento da Internet (LIU et al., 2013a). Com o intuito de fornecer acesso mais rapidamente aos serviços, os fabricantes adotaram diversas soluções diretamente relacionadas com o mecanismo de encaminhamento empregado no *hardware*.

FIB distribuída

Em cada placa de rede, ou interface é armazenada uma cópia completa da FIB de maneira que a decisão de encaminhamento seja realizada de forma independente (MOHAMED et al., 2012). Após o processador fazer a cópia da FIB para a interface (*download*), as solicitações de processamento ocorrem com menor frequência, como por exemplo no caso do destino não ser conhecido e um pacote ICMP de notificação tiver que ser enviado.

No núcleo da Internet a alta demanda por desempenho tornou necessário a utilização de roteadores com arquitetura de FIB distribuída (MOHAMED et al., 2012) que se diferencia da arquitetura centralizada porque os elementos do plano de dados e do plano de controle são executados em dispositivos de *hardware* diferentes (Figura 11).

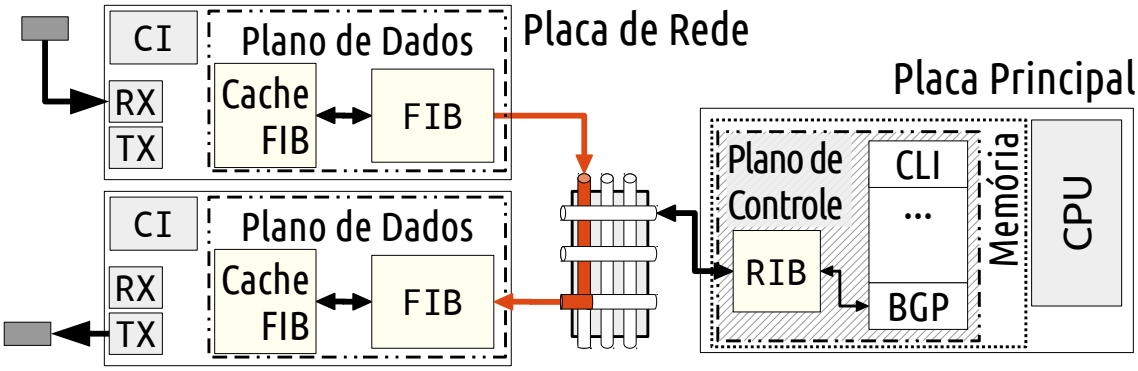


Figura 11 – Arquitetura FIB distribuída.

2.4.2 Protocolo baseado na seleção da FIB

A solução apresentada por Mohamed et al. reduz a FIB com a remoção de prefixos nas interfaces do roteador com arquitetura distribuída. A hipótese neste trabalho foi verificada com a transferência para a FIB apenas dos prefixos que podem ser alcançados através dos roteadores diretamente conectados. Desta forma, uma listagem reduzida é gravada na *cache* de cada interface ao invés de todos os destinos assinalados na RIB. Para isso, foi usado o campo reservado para os filtros de saída de rotas (ORF, *Outbound route Filter*) no protocolo BGP-4 (CHEN; REKHTER, 2008).

O ORF é um mecanismo que permite ao roteador que transmite a mensagem BGP informar quais rotas os receptores devem descartar. Os roteadores que a recebem aplicam esses filtros, além daqueles que estão configurados localmente.

A ideia é o preenchimento da FIB de cada interface apenas com os prefixos, cujos próximos saltos podem ser alcançados através do respectivo enlace físico. A proposta contribui para o problema do crescimento da FIB atuando em duas métricas temporais consideradas críticas na obtenção de uma Internet mais rápida e confiável para a quantidade crescente de usuários (MANRAL et al., 2005; BERKOWITZ et al., 2005):

Convergência da rede: A perda do acesso até que a rede convirja pode representar enormes prejuízos para a sociedade, como por exemplo os prejuízos financeiros resultantes da perda de conectividade das bolsas de valores momentos antes do fechamento (MOHAMED et al., 2012).

Latência: A manutenção na FIB de um menor número de prefixos torna o mecanismo de encaminhamento mais eficiente, pois a busca do prefixo ocorre mais rapidamente (TROTTER, 2001).

2.5 Software Defined Network (SDN)

A arquitetura da Internet foi concebida de forma a permitir independências entre os protocolos usados na infraestrutura física daqueles protocolos que provêm serviços às aplicações. Desta forma, novas tecnologias de *hardware* ou *software* poderiam ser acrescentadas mantendo a compatibilidade com o todo. O projeto define que os protocolos prestam serviços entre si, o que possibilita classificá-los como protocolos de baixo nível e protocolos de alto nível (COMER, 2006).

A interligação física entre os dispositivos é obtida com protocolos de baixo nível, como por exemplo, o Ethernet (PLUMMER, 1982). Estes protocolos fornecem o serviço de gerenciamento da transmissão pelo canal de comunicação para os protocolos de alto nível, que formam a pilha do TCP/IP. O protocolo IP fornece a interligação lógica entre os computadores pela determinação das rotas à serem seguidas no encaminhamento dos pacotes. Este serviço é prestado aos protocolos da camada de transporte (por exemplo, o TCP ou UDP), que por sua vez realizam a função de comunicação fim-a-fim para os protocolos de aplicação na origem e no destino.

Esta interligação entre os protocolos obteve grande sucesso, proporcionando aos desenvolvedores uma arquitetura em que podiam focar esforços nos aplicativos executados na origem e no destino. Em contrapartida, surgiu o fenômeno chamado “Ossificação da Internet” (MOREIRA et al., 2009), que caracteriza a dificuldade de alteração nos protocolos.

Desta constatação surgiram propostas para novas arquiteturas que são classificadas como puristas ou pluralistas (MOREIRA et al., 2009). Os puristas propõem a continuidade do modelo atual com uma única arquitetura uniforme, porém flexível o suficiente para anteder aos novos requisitos e aplicações. Os pluralistas, por outro lado, defendem a ideia de diversas arquiteturas de redes coexistindo em paralelo. Ambas abordagens consideram que a rede deve apresentar capacidade de adaptação para situações futuras.

Entre as motivações para adoção da SDN, a demanda por novas tecnologias e o aumento do número de usuários exemplifica a situação em que o crescimento da Internet ocorre mais rapidamente que a pesquisa e a produção de equipamentos adequados (HE, 2005). Além disso, o risco do investimento em tecnologias que podem ficar obsoletas rapidamente motivam o uso dos equipamentos sustentáveis empregados na arquitetura de rede definida por *software* (SDN, *Software Defined Network*). A interface de programação (API, *Application Programming Interface*) permite alterar os elementos do plano de controle, enquanto o plano de dados é especificado pelo fabricante no modelo do equipamento (MCKEOWN et al., 2008). Isso viabiliza a substituição progressiva dos equipamentos em redes híbridas, que empregam roteadores de diferentes arquiteturas e mesmo protocolo de enlace.

Um servidor de rede, denominado controlador OpenFlow, é responsável por coordenar as operações de todos os *switches* através de instruções enviadas para os dispositivos que fazem o encaminhamento dos pacotes. Esses dispositivos também sofrem influências no aumento da tabela de encaminhamento dos dados (FIB) quando empregam a mesma tecnologia que os roteadores da Internet (KANAUMI et al., 2012).

Na dissertação (SÁNCHEZ, 2013) é demonstrada a viabilidade da implementação do S-VA em redes OpenFlow (MCKEOWN et al., 2008). O algoritmo faz a coleta da RIB, calcula os prefixos virtuais de acordo com o número de participantes e faz a distribuição das regras de agregação (Seção 2.3) a partir do controlador NOX.

3 Solução Proposta

A proposta deste trabalho para reduzir a FIB tem como principais referências as pesquisas que empregam algoritmos de agregação na estrutura de dados (LIU et al., 2013b; UZMI et al., 2011) e filtragem dos registros quando existem múltiplas FIB (MOHAMED et al., 2012). Neste capítulo é explicado como estes estudos foram unificados de maneira a obter a filtragem em uma única FIB e agregar prefixos com outras combinações além do endereço do próximo salto (*next hop*) e a sinalização da melhor rota na RIB.

Na Seção 3.1 é explicado como são obtidos os dados necessários para a configuração da FIB. A arquitetura do algoritmo é apresentada na Seção 3.2 e o detalhamento dos seus módulos na Seção 3.3. Por fim, na Seção 3.4 é apresentado o comparativo com outras propostas e as considerações sobre uma unificação com a proposta apresentada.

3.1 Descrição da FIB

A configuração da FIB é feita com a gravação do prefixo da rede de destino, a interface usada para enviar os pacotes pelo enlace e o endereço IP do roteador de destino. Isto pode ser feito com a configuração estática na qual o operador digita cada rota, ou na configuração dinâmica que é obtida com *softwares* que monitoram os anúncios dos protocolos de roteamento (BGP ou OSPF) e alteram os registros de acordo com as mudanças da topologia da rede.

Nos *computadores pessoais*, uma configuração manual pode ser feita com comandos do sistema operacional, como o `ifconfig` no Linux. As suítes de roteamento são *softwares* que fazem dinamicamente a manutenção na tabela de informação de roteamento (RIB, *Routing Information Base*) de todas as rotas e selecionam a melhor rota para ser usada na FIB. O BIRD e o Quagga são exemplos de suítes de roteamento que possuem

outras funcionalidades, como a interface de console para controle do ambiente e comandos para estabelecer caminhos redundantes obtendo tolerância em situações de falhas (FARIAS et al., 2013).

Nos roteadores tradicionais, cada fabricante implementa o *software* controlador que disponibiliza opções de gerenciamento e manutenção dos elementos do plano de controle e de dados (CISCO Systems, 2012).

3.1.1 Roteadores Adjacentes

Numa topologia de rede, apenas os roteadores diretamente conectados entre si podem ser usados como próximo salto um do outro, de maneira que o mecanismo de roteamento correlacione a identificação da interface de saída e o endereço do próximo salto usada na FIB (TROTTER, 2001). Estas informações, juntamente com o prefixo da rede de destino, possibilitam o encapsulamento do pacote no quadro (*framming*) e o envio deste pelo enlace correto (BAKER, 1982).

Para "montar" o quadro é necessário o endereço físico do dispositivo associado ao IP de próximo salto registrado na FIB. Este endereço da interface no roteador adjacente é obtido com o protocolo de resolução de endereços (ARP, *Address Resolution Protocol*). Um pacote (ARP *request*) é enviado por difusão para todos os dispositivos diretamente conectados solicitando o reconhecimento do endereço IP e a resposta obtida é armazenada temporariamente na tabela ARP *cache*, onde estão listados os endereços lógicos e os respectivos endereços físicos. Esta tabela é verificada toda vez que é necessária a "tradução" do endereço IP para o endereço físico, evitando o envio de novos pedidos (PLUMMER, 1982). Com estes dois endereços o quadro pode ser montado e enviado para o enlace.

A Figura 12 exemplifica este procedimento com a criação de um quadro a partir das informações disponíveis na FIB. Na topologia, RA e RB são roteadores que ligam as redes "Rede-1" e "Rede-2" usando como caminho intermediário o roteador RN. Quando

um pacote com destino a "Rede-2" chega na interface *eth1* de RN, é feita a verificação na FIB de qual o endereço IP do próximo salto e a interface de saída que deverá ser usada.

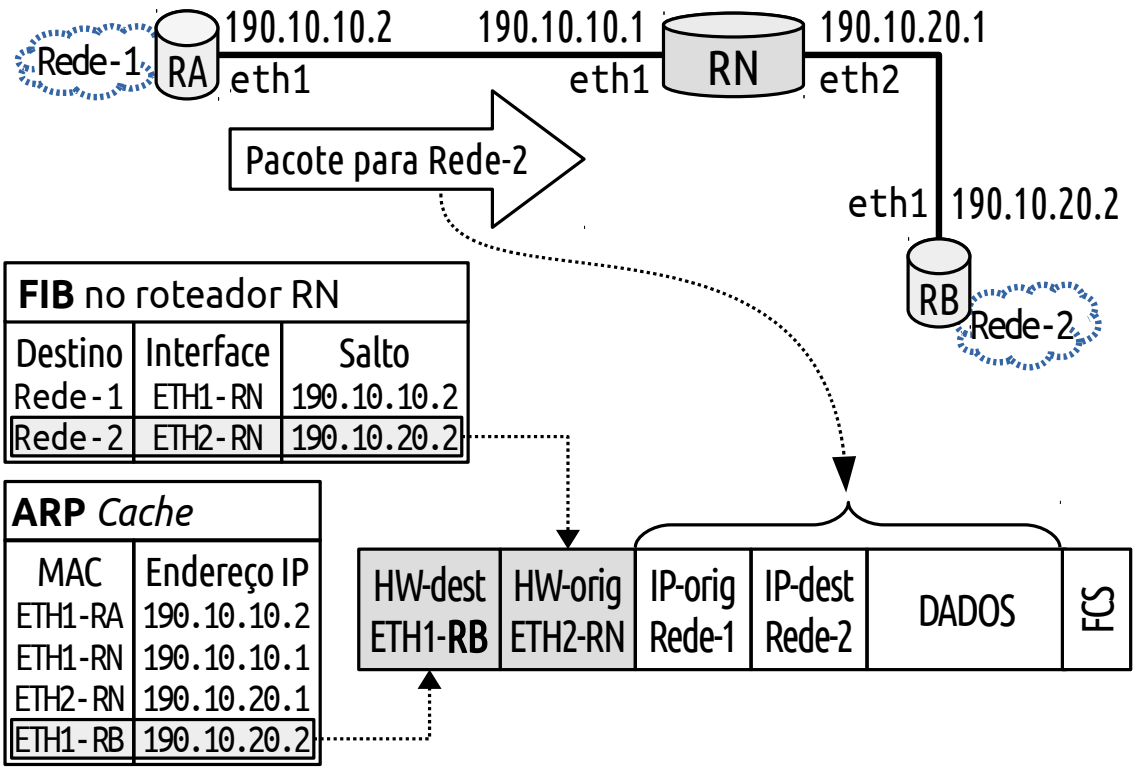


Figura 12 – Informações usadas na criação do quadro.

Quando o protocolo de roteamento recebe o anúncio de um endereço IP usado como próximo salto que não está diretamente conectado, como por exemplo em anúncios de outros Sistemas Autônomos obtidos *via iBGP*, o mecanismo de roteamento determina quais roteadores adjacentes podem ser usados como *next hop*. Caso não seja possível determinar pelo menos um salto diretamente conectado, é enviada uma mensagem de gerenciamento informando que este roteador não pode ser intermediário para a rede anunciada (ICMP *Bad Source Route*) (BAKER, 1982).

3.1.2 *Lookup Nexthop*

A FIB possui apenas uma rota para o prefixo da rede de destino. Isto ocorre porque o mecanismo de busca do próximo salto (*Lookup Nexthop*) é usado para procurar a primeira

correspondência entre o endereço IP de destino do pacote e os prefixos das redes de destino que estão registrados na FIB. Esta verificação é realizada primeiramente nos endereços de *host* /32 e depois nos prefixos de rede em ordem decrescente do tamanho da máscara. Quando nenhuma correspondência é encontrada, a rota com prefixo de rede zero (`0.0.0.0/0`) é usada (BAKER, 1982). O emprego desta "rota padrão" (*default route*) garante que o pacote não será descartado caso haja um rota específica na FIB.

Quando a primeira correspondência é encontrada, a operação de *Lookup Nexthop* é finalizada, por isso na RIB somente um caminho para o destino é marcado como ativo (melhor caminho) (TROTTER, 2001). Caso fossem marcados caminhos alternativos, estes seriam enviados para a FIB resultando no desperdício de memória, pois apenas uma correspondência é obtida. Em situações que são necessárias rotas diferentes para o mesmo prefixo de rede, como por exemplo no balanceamento de carga, os protocolos de rede usam rotas multi-caminho com custo igual (ECMP, *Equal cost multi-path*) criadas para este propósito (HOPPS, 2000).

A configuração padrão armazena todas as rotas na tabela de roteamento principal, porém elas podem ser agrupadas ou referenciadas quando fazem parte de outras tabelas de roteamento. As rotas obtidas por diferentes protocolos de roteamento, as rotas estáticas e as rotas diretamente conectadas fazem parte de grupos diferenciados na tabela principal, enquanto as rotas ECMP descritas anteriormente e as que usam a interface de saída como identificador do próximo salto estão em outras tabelas. Em cada tabela apenas uma rota para o prefixo de destino pode ser marcada como o melhor caminho, por isso o balanceamento de carga é obtido com rotas registradas em diferentes tabelas.

3.1.3 Identificadores do próximo salto

A FIB pode ser configurada usando como identificador o endereço IP (Tabela 6a) ou a interface de saída (Tabela 6b). Não há mudança nos campos presentes na FIB, porém o mecanismo de *Lookup Nexthop* não é executado quando a interface é usada como

identificador de próximo salto. Isto se aplica apneas no caso de interfaces de saída para enlaces ponto a ponto.

Destino	**Próximo Salto**
72.208.0.0/12	190.10.10.2
133.10.0.0/16	190.10.20.2

(a)

Destino	**Interface**
72.208.0.0/12	eth1
133.10.0.0/16	eth2

(b)

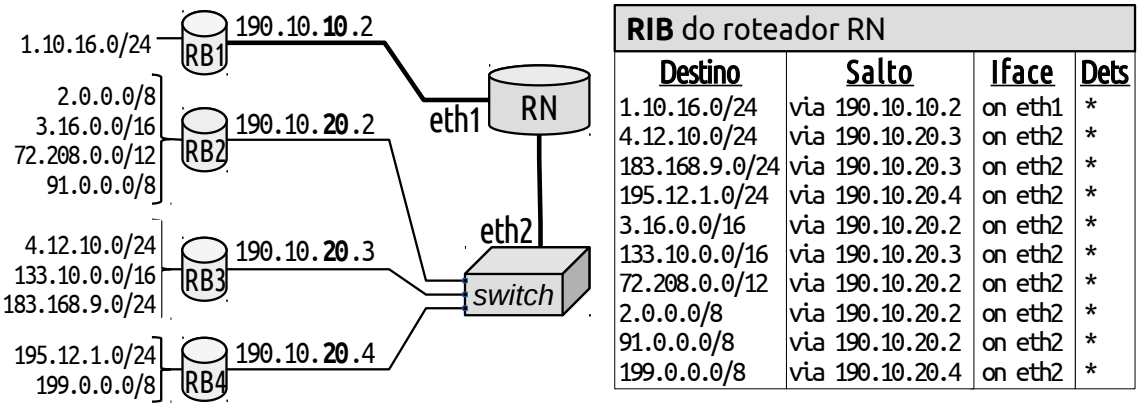
Tabela 6 – Identificadores usados para configurar a FIB.

Para encaminhar os pacotes, a identificação da interface e o endereço IP no destino são necessários para a criação do quadro enviado (Figura 12). Se for usada o IP como identificador, o mecanismo de encaminhamento assume que o próximo salto está diretamente conectado e o endereço físico no destino pode ser obtido com uma solicitação ARP *request* para o endereço físico de *broadcast* (`FF:FF:FF:FF:FF:FF`) (PLUMMER, 1982). Para obter o encaminhamento entre os saltos intermediários até um prefixo de destino que não estejam no mesmo enlace, é necessário que a tabela ARP tenha sido configurada com o endereço do destino associado ao próximo salto. Isso pode ser feito com a configuração manual da tabela ARP ou com a utilização do *proxy* ARP.

Essa solução usando como identificação o endereço IP não é indicada quando uma *bridge* é usada como intermediário para um grupo de roteadores (Figura 13) (TROTTER, 2001) porque seria necessário configurar a tabela ARP manualmente, o que não é uma solução escalável.

3.2 Modelo Proposto

A solução apresentada configura a FIB usando os registros da RIB sempre que a mesma é atualizada (Figura 14). Para isto, duas informações são usadas: o prefixo da rede de destino e a interface usada para enviar os pacotes para o próximo salto da rota. A redução da FIB é obtida calculando-se um novo prefixo para representar exclusivamente um agrupamento



RIB do roteador RN

Destino	Salto	Iface	Dets
1.10.16.0/24	via 190.10.10.2	on eth1	*
4.12.10.0/24	via 190.10.20.3	on eth2	*
183.168.9.0/24	via 190.10.20.3	on eth2	*
195.12.1.0/24	via 190.10.20.4	on eth2	*
3.16.0.0/16	via 190.10.20.2	on eth2	*
133.10.0.0/16	via 190.10.20.3	on eth2	*
72.208.0.0/12	via 190.10.20.2	on eth2	*
2.0.0.0/8	via 190.10.20.2	on eth2	*
91.0.0.0/8	via 190.10.20.2	on eth2	*
199.0.0.0/8	via 190.10.20.4	on eth2	*

Figura 13 – Endereço IP como identificador do próximo salto.

de endereços. Este prefixo funciona como um filtro para apontar a interface de saída que deve ser usada para atingir os prefixos de destino contidos neste agrupamento.

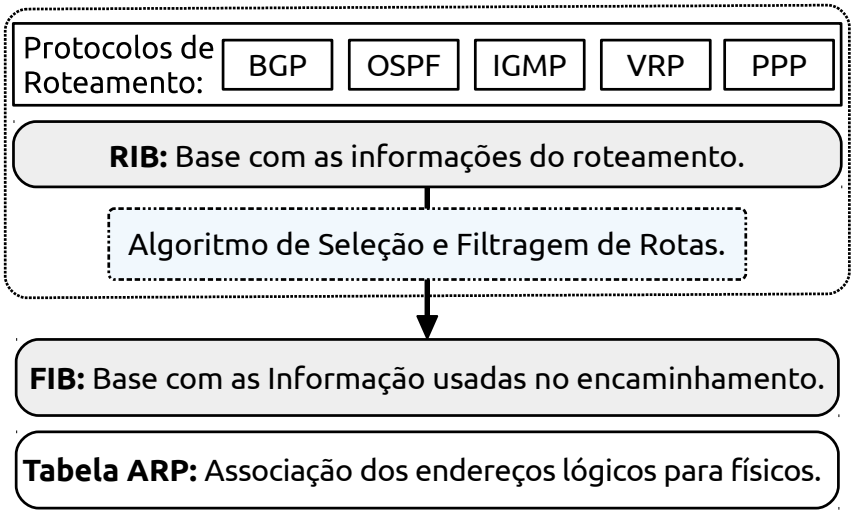


Figura 14 – Ligação entre o algoritmo e os elementos de rede.

Desta forma, procura-se criar agrupamentos de redes de destino cujos endereços de próximo salto estejam conectados através de um mesmo enlace (ou interface de rede). A ideia consiste em se obter o menor número possível de agrupamentos e de representar unicamente cada agrupamento por um mesmo prefixo de rede.

Esta abordagem considera dois aspectos importantes de ordem prática: (i) que a distribuição geográfica das faixas de IP pelo mundo gera situações nas quais diversos blocos de rede, anunciados por diferentes AS de uma mesma região, possuam faixas de IP

relativamente próximas umas das outras e uma mesma rota de saída; (ii) que a interligação entre roteadores se dá através de enlaces ponto-a-ponto.

Este último aspecto é considerado no algoritmo proposto de maneira a aumentar no nível de redução da FIB, o que se torna possível quando não se faz necessário considerar o endereço de próximo salto das rotas, apenas a interface de saída que conduz a este próximo salto.

3.3 Descrição dos Módulos do Algoritmo

O algoritmo gerencia duas estruturas de dados para determinar a agregação. A primeira é usada para armazenar os dados coletados e na segunda estão os resultados obtidos. Diferentes módulos de sub-rotinas manipulam estas tabelas, por isso a arquitetura do algoritmo foi dividida em três etapas (Figura 15).

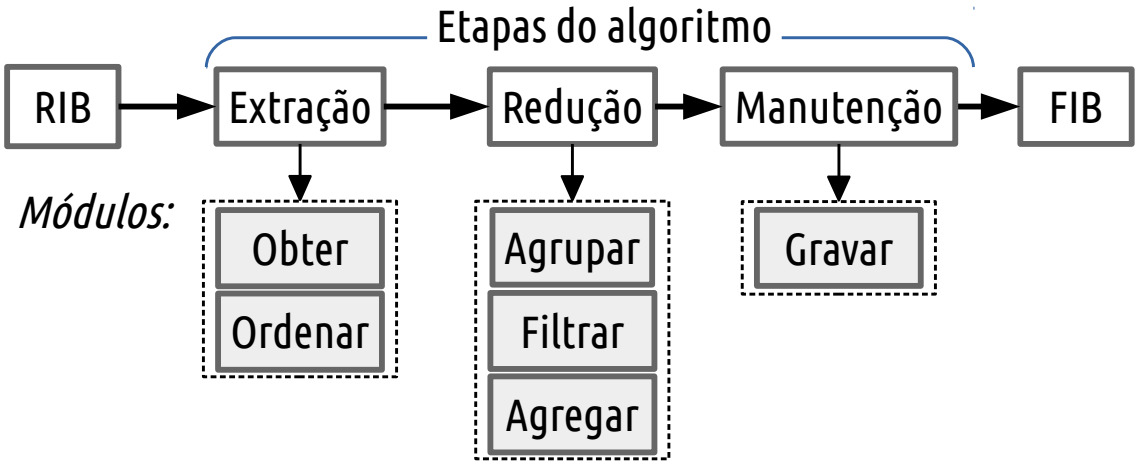


Figura 15 – Etapas do algoritmo.

EXTRAÇÃO: Esta é a primeira etapa do algoritmo que resulta na criação da tabela usada nas etapas seguintes. Os registro são selecionados na RIB e estão indexados pelo prefixo da rede de destino. Esta etapa é composta por dois módulos: *obter* e *ordenar*. O *script* usado na implementação destes módulos é listado no Apêndice A.

REDUÇÃO: Nesta segunda etapa os prefixos da tabela extraída da RIB são filtrados e agregados de forma a gerar o conjunto de prefixos exclusivos usados na FIB reduzida. Esta etapa é composta por três módulos: *agrupar*, *filtrar* e *agregar*. A rotina no Apêndice B resulta na FIB reduzida, que é armazenada no arquivo usado na próxima etapa.

MANUTENÇÃO: Nesta última etapa as informações na tabela *agregada* são gravadas na FIB. Nesta etapa também é prevista a inclusão de novos prefixos no caso de convergência da rede. Esta etapa é composta por um único módulo: *gravar*. O Apêndice C é o *script* usado nesta etapa.

3.3.1 Módulo 1: Obtenção dos Dados

O primeiro módulo (Figura 16) utiliza campos da RIB para criar vetores assimétricos que serão empregados nas etapas seguintes. A *chave* do vetor são os registros do campo DESTINO e o *valor* do vetor são os registros indicadores da INTERFACE. A alternativa de usar o PRÓXIMO SALTO ao invés da INTERFACE é realizada neste ponto. Os registros na RIB são selecionados de acordo com a presença do asterisco no campo DETALHES. Este *flag* é usado como indicativo da escolha da melhor rota, por isso é usado para indicar a entrada que será carregada no vetor.

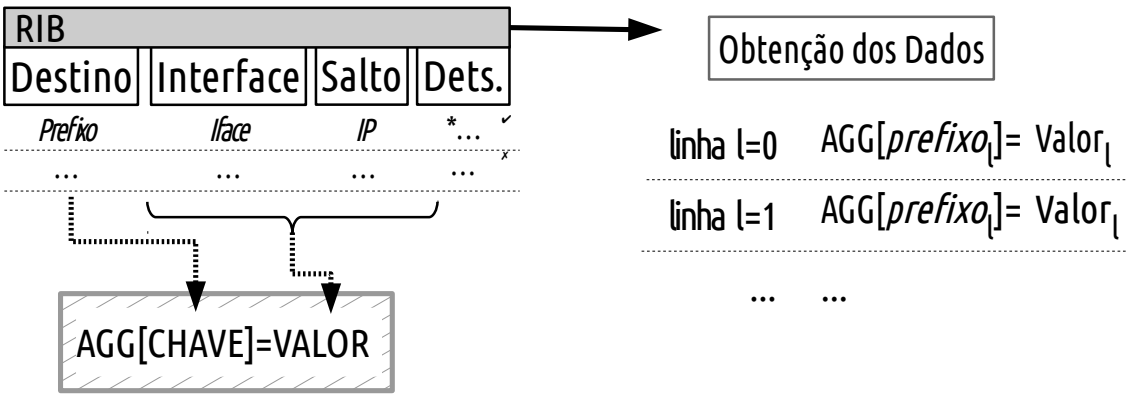


Figura 16 – Modulo de obtenção dos dados da RIB.

Para exemplificar os resultados obtidos nas operações será usada a configuração

do roteador da Figura 17. Nesta topologia os diferentes sistemas autônomos são usados como caminho para as redes cujos prefixos estão listados ao lado da topologia. Essa é uma ligação multi salto que possibilita o balanceamento de carga ou a tolerância à falhas. No caso da RIB listada na figura, apenas um registro associado ao prefixo de rede é marcado com o *flag* da indicação do melhor caminho.

O conteúdo de um vetor onde os prefixos de destino são as chaves e as interfaces de saída dos pacotes e os *valores* é exemplificado na Tabela 7a. Outras variáveis registradas no campo detalhes podem ser usadas para selecionar as entradas da RIB obtendo outros resultados além da redução. Por exemplo, caso sejam selecionadas entradas associadas aos *AS Path* 20 ou 30, o algoritmo configura na FIB um caminho redundante até a rede 183.168.9.0/24 usando as interfaces *eth2* e *eth3* (Tabela 7b).

RIB do roteador RN

Destino	Salto	Iface	Dets
...	...	...	...
72.208.0.0/12	15.0.0.1	eth1	AS15
72.208.0.0/12	20.0.0.2	eth2	* AS20
133.10.0.0/16	15.0.0.1	eth1	* AS15
133.10.0.0/16	30.0.0.3	eth3	AS30
183.168.9.0/24	20.0.0.2	eth2	AS20
183.168.9.0/24	30.0.0.3	eth3	* AS30
...	...	...	...

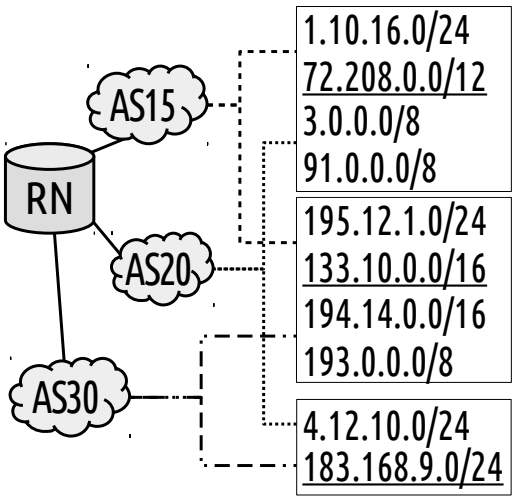


Figura 17 – Rede de exemplo com múltiplos caminhos

...	
AGG[183.168.9.0/24] =	eth3
AGG[133.10.0.0/16] =	eth2
AGG[72.208.0.0/12] =	eth2

(a)

AGG[183.168.9.0/24] =	eth2
AGG[183.168.9.0/24] =	eth3
AGG[133.10.0.0/16] =	eth2
AGG[72.208.0.0/12] =	eth2

(b)

Tabela 7 – Vetores assimétricos com as interfaces.

3.3.2 Módulo 2: Ordenação dos Vetores

Neste segundo módulo (Figura 18) é realizada a ordenação crescente do vetor. Para isso, a sequência é ordenada várias vezes, uma para cada octeto do prefixo. O primeiro octeto é usado na primeira passagem e os três octetos seguintes são usados nas sequências seguintes, como elementos de diferenciação no registro. Após o último octeto ser empregado, o valor decrescente da máscara da rede é empregado. Dessa forma, o número binário equivalente do prefixo é organizado em ordem crescente.

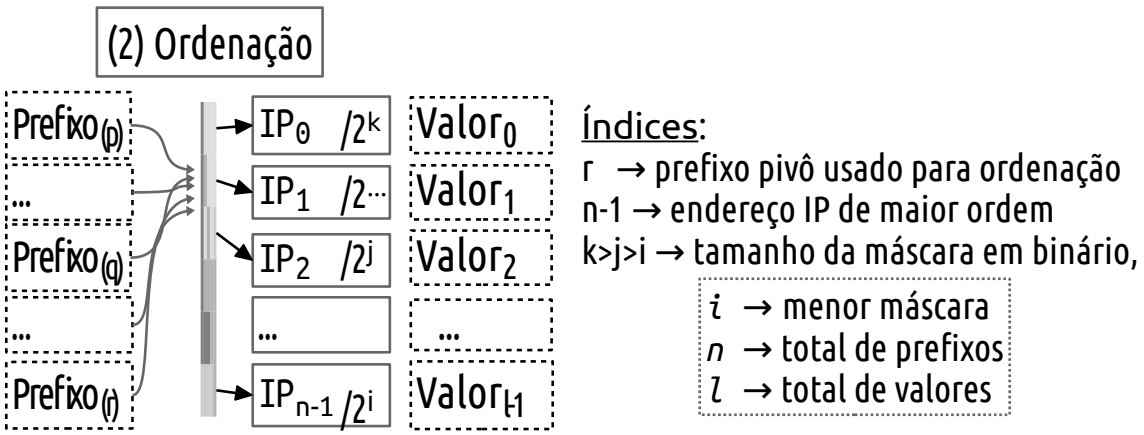


Figura 18 – Modulo para ordenação do vetor assimétrico.

Esta forma de ordenar tem como base o algoritmo de ordenação estável *Mergesort*, que ordena a sequência várias vezes. A complexidade no pior caso é `O(n log n)` e tem baixa carga de processamento quando comparada com outras técnicas de ordenação (SINHA; ZOBEL, 2004), porque apenas parte do registro pode ser necessário para obter o resultado. No exemplo listado na Tabela 8a, a ordenação de todos os vetores é obtida com a indexação usando o primeiro octeto resultando na Tabela 8b.

3.3.3 Módulo 3: Formação do Agrupamento

O propósito deste módulo é delimitar a listagem, separando os vetores que possuem o mesmo *valor*. Isto é feito para possibilitar que a filtragem, realizada no próximo módulo, seja executada em agrupamentos de entrada de mesmo *valor*. Como representado na Figura

AGG[**1**.10.16.0/24] =	eth2
AGG[**4**.12.10.0/24] =	eth2
AGG[**183**.168.9.0/24] =	eth3
AGG[**195**.12.1.0/24] =	eth1
AGG[**133**.10.0.0/16] =	eth2
AGG[**194**.14.0.0/16] =	eth1
AGG[**72**.208.0.0/12] =	eth2
AGG[**3**.0.0.0/8] =	eth2
AGG[**91**.0.0.0/8] =	eth2
AGG[**193**.0.0.0/8] =	eth1

(a)

AGG[1.10.16.0/24] =	eth2
AGG[3.0.0.0/8] =	eth2
AGG[4.12.10.0/24] =	eth2
AGG[72.208.0.0/12] =	eth2
AGG[91.0.0.0/8] =	eth2
AGG[133.10.0.0/16] =	eth2
AGG[183.168.9.0/24] =	eth3
AGG[193.0.0.0/8] =	eth1
AGG[194.14.0.0/16] =	eth1
AGG[195.12.1.0/24] =	eth1

(b)

Tabela 8 – Ordenação dos vetores.

19, a identificação do grupo corresponde ao índice do *valor* no vetor agrupado *GRP*. O prefixo da rede de destino é a *chave* deste vetor.

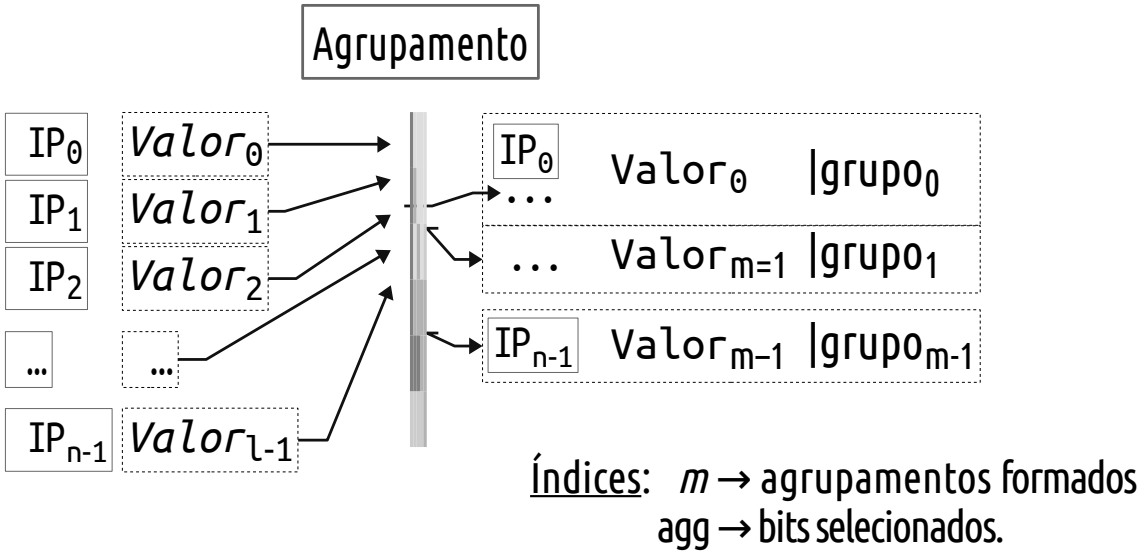


Figura 19 – Módulo de formação do agrupamento.

3.3.4 Módulo 4: Filtragem dos agrupamentos

Para cada grupo formado, todos os vetores são descartados exceto o primeiro (`IPx`) e o último (`IPy`). Este processo de refinamento representado Figura 20a é equivalente à redução obtida com técnicas empregadas em roteadores que possuem uma FIB em cada interface de saída (Seção 2.4). Porém, o módulo de agregação (Subseção 3.3.5) tem

Prefixo	Interface	Grupos	
3.0.0.0/8	eth2	GRP(3.0.0.0/8)	=1
4.12.10.0/24	eth2	GRP(4.12.10.0/24)	=1
72.208.0.0/12	eth2	GRP(72.208.0.0/12)	=1
91.0.0.0/8	eth2	GRP(91.0.0.0/8)	=1
133.10.0.0/16	eth2	GRP(133.10.0.0/16)	=1
183.168.9.0/24	eth3	GRP(183.168.9.0/24)	=2
193.0.0.0/8	eth1	GRP(193.0.0.0/8)	=3
194.14.0.0/16	eth1	GRP(194.14.0.0/16)	=3
195.12.1.0/24	eth1	GRP(195.12.1.0/24)	=3

Tabela 9 – Criação dos agrupamento por interface.

que ser executado de forma a calcular a máscara de rede, possibilitando que a lógica de encaminhamento funcione corretamente.

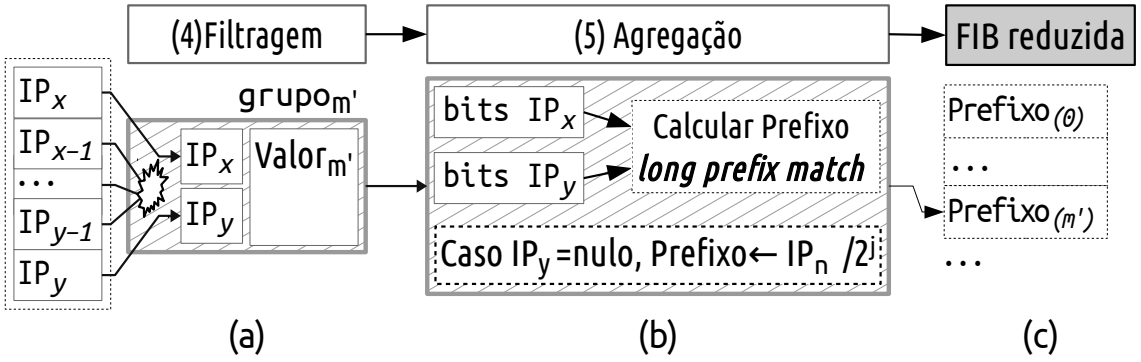


Figura 20 – Módulos de (a) Filtragem e (b) Agregação. (c) FIB reduzida.

Agrupamentos formados com duas ou mais entradas resultam em dois prefixos associados com uma mesma interface. Em contra partida, agrupamentos que têm um único vetor apresentam no segundo prefixo (`IPy`) o valor nulo. Por isso, a quantidade de entradas associada a um mesmo agrupamento é um critério de verificação da eficiência do algoritmo para redução da FIB. Quanto maior a quantidade em um mesmo agrupamento, maior será a taxa de redução obtida na filtragem. O pior caso é observado na ocorrência de apenas uma entrada no grupo. Na Tabela 10a o prefixo `183.168.9.0/24` representa uma única entrada no grupo, consequentemente representa o prefixo do agrupamento na Tabela 10b.

1.10.16.0/24 =	eth2
3.0.0.0/8 =	eth2
4.12.10.0/24 =	eth2
72.208.0.0/12 =	eth2
91.0.0.0/8 =	eth2
133.10.0.0/16 =	eth2
183.168.9.0/24 =	eth3
193.0.0.0/8 =	eth1
194.14.0.0/16 =	eth1
195.12.1.0/24 =	eth1

(a)

GRP1	1.10.16.0/24 =	eth2
	133.10.0.0/16 =	eth2
GRP2	183.168.9.0/24 =	eth3
GRP3	193.0.0.0/8 =	eth1
	195.12.1.0/24 =	eth1

(b)

Tabela 10 – Filtragem dos agrupamentos.

3.3.5 Módulo 5: Agregação dos prefixos

A filtragem realizada no módulo anterior possibilita que o cálculo do maior prefixo comum existente entres os pares formados no agrupamento (LPM, *Longest Prefix-Match*) seja usado como prefixo agregador. O menor prefixo comum (SPM, *Shortest Prefix Match*) não é usado porque resultaria em prefixos agregadores iguais associados a interfaces diferentes, tornando necessário uma sub-rotina de desambiguação para a determinação de um único prefixo exclusivo.

Primeiramente é verificada a quantidade de prefixos que estão no grupo. No caso de haver apenas um, o cálculo de agregação não é realizado, e o prefixo e a interface são usados diretamente para a configuração da FIB reduzida. Nos outros casos, apenas o endereço IP dos pares são usados e as máscaras suprimidas. Esta condição é realizada apenas nos casos que os agrupamentos tiveram entradas suprimidas, pois uma nova máscara será calculada para representar todos os prefixos do agrupamento. Este método de agregação funciona mesmo nos casos de rotas para *hosts* estarem na RIB. Por exemplo, o endereço `10.0.10.1/16` de um *host* posicionado entre os prefixos `10.0.0.0/24` e `11.0.0.0/8` na etapa de ordenação é suprimido na etapa de filtragem, resultando um prefixo agregador que inclui todos os endereços IP entre `10.0.0.0` e `11.0.0.0`.

Para determinar o prefixo agregador é necessário a descoberta de uma máscara de rede que o represente. A função coincidência é empregada para o cálculo da máscara de rede usada no prefixo agregador. Para isso, o axioma lógico 3.1 é usado em cada um dos *bits* de *IPx* e *IPy*. O tamanho da máscara é determinado na posição cujo resultado é `0`.

$$(IPx \wedge\ IPy) \vee (\neg IPx \wedge \neg IPy) \tag{3.1}$$

Um exemplo da operação da Equação 3.1 é mostrado na Tabela 11. Dado os endereços IPx e IPy o valor da máscara é determinado com a operação lógica bit a bit até que o primeiro *zero* seja encontrado. Esta máscara aplicada no endereço IPx ou no IPy resulta no prefixo agregador.

$IPx = 193.0.0.0 =$	`11000001.00000000.00000000.00000000`
$IPy = 195.12.1.0 =$	`11000011.00001100.00000001.00000000`
$(IPx \bullet IPy) + (\overline{IPx} \bullet \overline{IPy}) =$	`111111--.--------.--------.--------`
Bits coincidentes	`6`
Máscara	**`110000`**`00.00000000.00000000.00000000`
Prefixo agregador	**`192.0.0.0/6`**

Tabela 11 – Exemplo do cálculo de máscara.

3.3.6 Módulo 6: Gravação dos prefixos na FIB

Neste trabalho, a gravação é o módulo implementa que efetiva a utilização dos prefixos.encontrados. A ideia é interferir com os outros processos que interagem com a tabela de encaminhamento, substituindo o conteúdo existente pelo resultado encontrado.

3.4 Comparação com outras tecnologias

O algoritmo de redução da FIB usando a filtragem em uma arquitetura centralizada foi concebido em módulos. Isto permite que as técnicas usadas nas pesquisas anteriores (Capítulo 2) sejam incorporadas, obtendo a adaptabilidade em outros cenários e a melhoria

da eficiência. Nesta seção, são apresentadas as comparações com estas técnicas e a melhoria que pode ser obtida com a incorporação das mesmas nos módulos do algoritmo.

3.4.1 Sumarizando anúncios de rotas com a seleção da FIB

O CIDR (*Classless Inter-Domain Routing*) pode ser usado nos roteadores da Internet para reduzir a quantidade de anúncios. Esta técnica usa o *Longest-Prefix Matching* para agregar os prefixos enviados pelo protocolo de roteamento. Porém, existem limitações quanto ao seu uso:

- A topologia da rede pode resultar na determinação incorreta da melhor rota.
- A mesclagem de todos os prefixos associados com uma interface pode resultar no anúncio indesejado de certos blocos de endereços.

Na ilustração da Figura 21a, dois grupos de roteadores estão posicionados em planos diferentes. Os roteadores de borda foram conectados de maneira a obter redundância entre o primeiro e o segundo grupo. A Figura 21b representa os blocos do ISP-A sumarizados em BA1 e BA2.

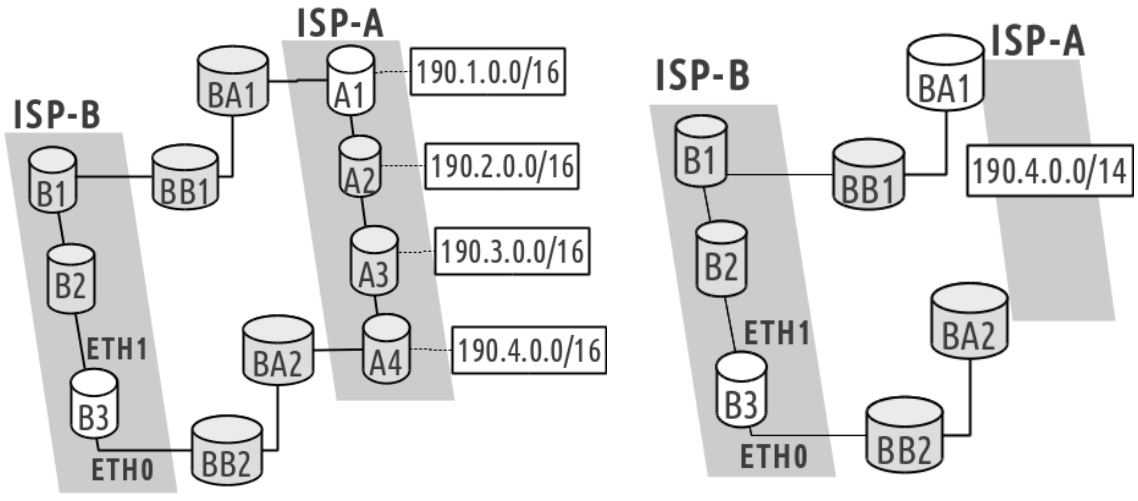


Figura 21 – Cenário com caminhos redundantes.

Para demonstrar a influência da topologia de rede sobre o CIDR, a tabela de encaminhamento do roteador B3 no ISP-B é listada. No caso de não haver a sumarização, a Tabela 12a indica que o melhor caminho para a rede $190.2.0.0/16$ é usando a saída pela interface *eth0*, enquanto que a configuração do CIDR nos dois roteadores de borda do ISP-A resulta como melhor caminho a interface *eth0*, como indicado na Tabela 12b. A comparação entre as tabelas demonstra que a melhor rota dos pacotes para A1 deveria ser enviado pela interface *eth1*. Este problema ocorre porque o mesmo prefixo sumarizado é anunciado pelos dois roteadores de borda, e a menor distância até este destino é através do roteador BB2, ou seja, $eth0$.

Destino	**Salto**	**Interface**
190.1.0.0/16	B2	eth1
190.2.0.0/16	BB2	eth0
190.3.0.0/16	BB2	eth0
190.4.0.0/16	BB2	eth0

(a)

Destino	**Salto**	**Interface**
190.4.0.0/14	BB2	eth0

(b)

Tabela 12 – Tabela de encaminhamento no roteador B3.

A solução é a configuração do anúncio de maneira a distinguir os caminhos. Porém, a forma como os prefixos estão distribuídos torna a tarefa mais complicada e menos eficiente. Por exemplo, a agregação de $190.2.0.0/16$ com $190.3.0.0/16$ resulta no prefixo $190.2.0.0/15$ que pode ser anunciado com $190.1.0.0/16$ e $190.4.0.0/16$. Assim, o roteador B3 pode escolher o melhor caminho usando as métricas definidas em sua configuração usando a proposta apresentada neste trabalho.

3.4.2 Comparação com as técnicas de agregação

Os mecanismos usados para agregação apresentados no Capítulo 2 usam algoritmos de busca TRIE, em estruturas formadas com todos os bits do prefixo, ou o algoritmo de busca otimizada PATRICIA em árvores compactadas (LIU et al., 2013b). A redução do tempo de agregação obtido com o *TRIE* em comparação ao *Longest-Prefix Matching* foi comprovado por Draves et al. (1999) com o ORTC.

O uso de estruturas em árvores pelo algoritmo desta dissertação tem que ser verificado quanto à melhora do desempenho geral. A simplicidade na comparação de registros (etapas de redução) e no cálculo booleano do *Longest-Prefix Matching* em dois IPs (etapa de manutenção) foi a principal motivação da escolha desta técnica, ao invés da eficiência obtida com o alto custo de processamento no uso do PATRICIA (MORRISON, 1968).

O ORTC foi usado como referência nos trabalhos seguintes. No SMALTA (UZMI et al., 2011) uma segunda tabela com os resultados da agregação é usada para reduzir a carga de processamento de toda a estrutura quando ocorrerem alterações de prefixos. Além deste, o FIFA (LIU et al., 2013b) utiliza três algoritmos para gerenciar a carga de processamento com a seleção do momento que a agregação é processada.

Com base nestas propostas, a redução do tempo na convergência da rede pode ser obtida com a implementação, na etapa de manutenção, de um mecanismo residente em memória (*daemon*) que sinalize os prefixos não agregados e faça o agendamento da etapa de redução. A diferença deste módulo adicional da proposta no SMALTA e no FIFA é a possibilidade da desambiguação nos agrupamentos de prefixos. Por exemplo, os prefixos podem ser inscritos diretamente na FIB até o próximo agendamento porque, quando ocorre a alteração da topologia, uma outra interface é usada para a saída dos pacotes.

Por exemplo, a Tabela 13 é uma possível solução da FIB para o roteador B3 ilustrado na Figura 21, caso o roteador de borda BB2 apresente defeito. A detecção da falha pode ser obtida com mensagens ORF (Subseção 3.4.1) ou através da RIB.

Destino	**Salto**	**Interface**
190.1.0.0/16	B2	eth1
190.2.0.0/16	B2	eth1
190.0.0.0/14	B2	eth1
190.0.0.0/13	BB2	eth0

Tabela 13 – FIB temporária em B3 na presença de falhas de BB2

3.4.3 Redes definidas por *software*

A Figura 22 ilustra como funciona uma arquitetura típica SDN. O controlador configura no cliente uma série de entradas na tabela de fluxo que são usadas na decisão de encaminhamento. Quando o *switch* recebe um pacote, ele deve compará-lo com a sua tabela de fluxo. Se o cabeçalho não for compatível com nenhuma das entradas, ele pode encaminhá-lo para o controlador, que toma uma decisão e responde com uma ação. Por exemplo, um pacote, ao chegar em uma interface, pode ser encaminhado para outra interface que será usada até o próximo salto ou descartado.

A determinação da interface de saída que será usada para o próximo salto do caminho pode ser obtida com elementos do plano de controle em ambiente virtualizado. Máquinas virtuais com suítes de roteamento podem ser usadas para armazenar as RIBs associadas aos *switches* do plano de dados. O resultado é a separação dos elementos do plano de dados dos elementos do plano de controle que estão contidos em máquinas virtuais.

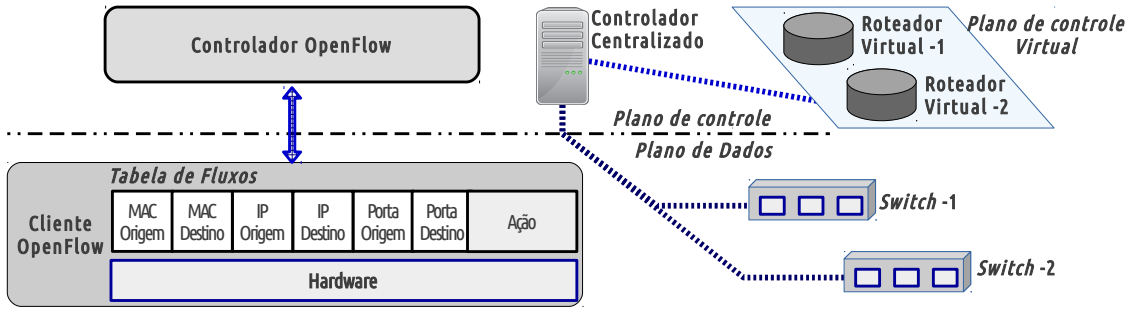


Figura 22 – Plataforma Virtual de Roteamento.

As redes definidas por *software* (SDN) representam uma vantagem na possibilidade de um gerenciamento centralizado, com níveis claros de abstração que permitem o desenvolvimento de sistemas de rede, independente de fabricante de *hardware*. Porém, a possibilidade de usar tecnologias como a virtualização (FARIAS et al., 2013) e sobreposição de redes (ARAÚJO et al., 2012) resulta em maior demanda de processamento da tabela de encaminhamento (FIB) do que os roteadores tradicionais (ROTTENSTREICH et

al., 2013).

A proposta nesta dissertação se adapta como uma solução porque incorpora a seleção dos registros para as interfaces dos roteadores distribuídos (Figura 11) (MOHAMED et al., 2012), o tunelamento entre roteadores agregadores (APR , Figura 9) usados no *Simple Virtual Aggregation* (RASZUK et al., 2012) e as conclusões da implementação em dispositivos OpenFlow (SÁNCHEZ, 2013).

A ideia é que cada controlador receba apenas os prefixos associados com as suas interfaces além do prefixo agregado usado pelos outros dispositivo na rota. Para isso, as etapas de redução e manutenção são alteradas para registrar a identificação do controlador e a interface usada, além do endereço do próximo salto usado. O resultado é a criação de uma FIB geral e uma FIB específica para cada *switch*. A Figura 23 representa os elementos de rede envolvidos.

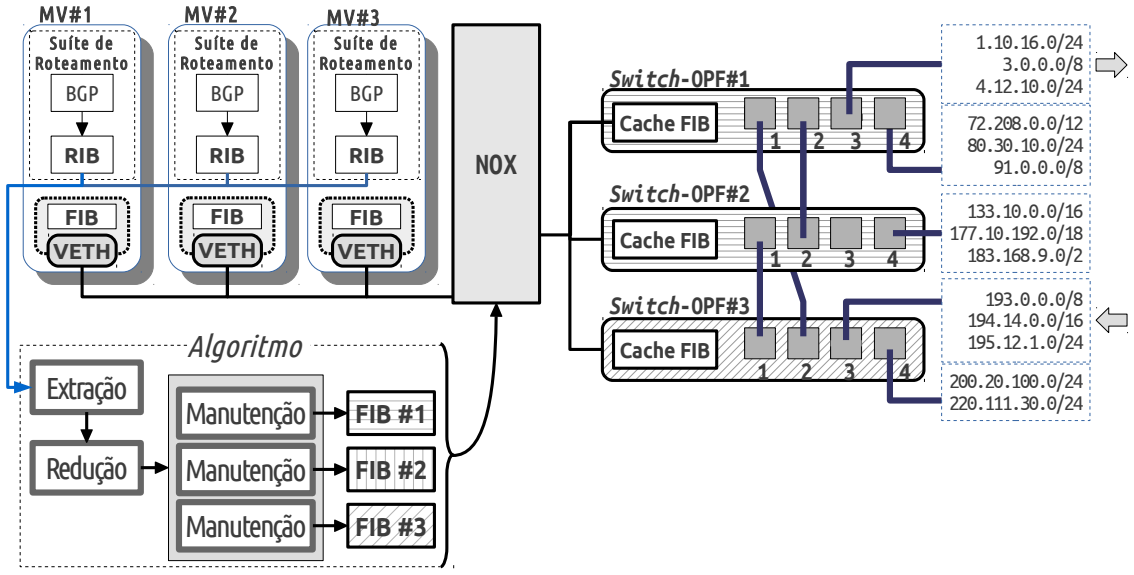


Figura 23 – Desenho do algoritmo para arquitetura SDN.

O algoritmo proposto é implementando junto aos outros elementos do plano de gerência, como o *software* de protocolo e o controlador (NOX), para ter acesso aos registros da RIB e enviar o resultado do processamento para cada um dos dispositivos. Na etapa de *extração de dados* os prefixos da RIB são coletados da forma usual, porém é

feito o acréscimo da identificação do dispositivo, além da interface. Na etapa de *redução* é obtido a agregação das FIBs usadas para programas os dispositivos OpenFlow. E a etapa de *manutenção* atua em paralelo com o NOX para programar e distribuir os prefixo.

Como exemplo, as Tabelas 14a, 14b, 14c e 14d são os resultados obtidos em cada etapa da implementação com os prefixos listados na Figura 23. Não são reapresentados os módulos de *obtenção* e *ordenação* (etapa de *extração*), por isso os prefixos já estão devidamente selecionados e ordenados. Os dados são agrupados (Tabela 14a), filtrados (Tabela 14b) e agregados (Tabela 14c) da forma usual. A diferença é a *FIB geral* (Tabela 14d) obtida com a repetição dos módulos *agrupar* e *agregar* usando os dados obtidos na *FIB específica* (Tabela 14c).

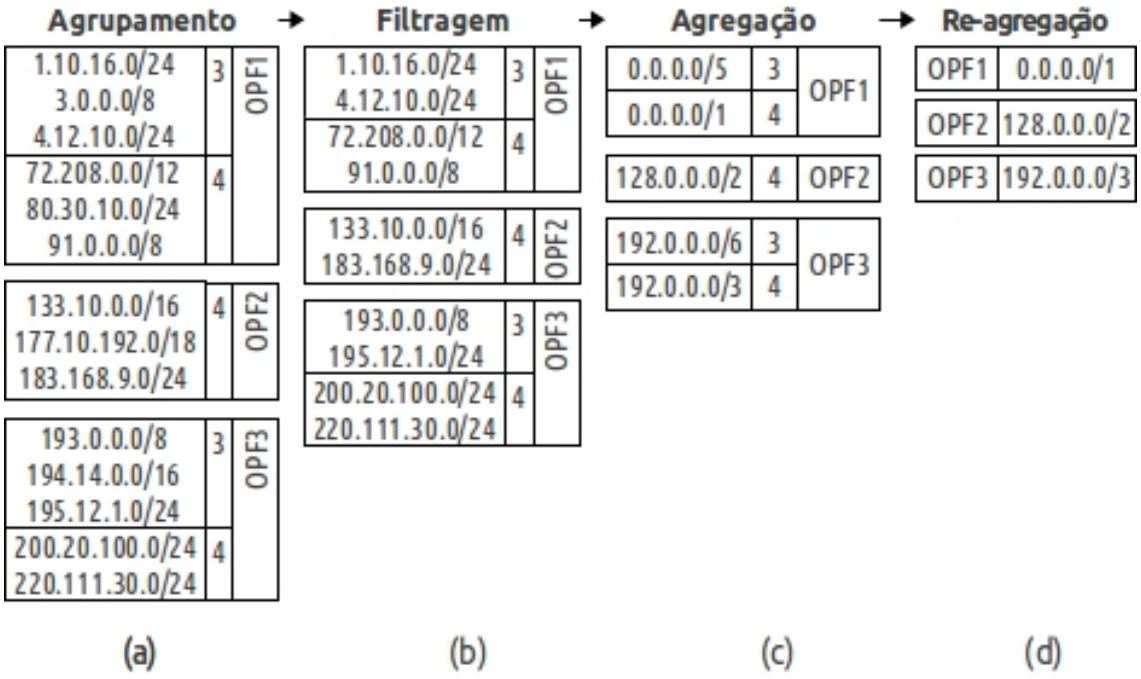


Tabela 14 – Resultados obtidos em *redução* e *manutenção* no OpenFlow.

Na etapa de *manutenção* os resultados são usados para refletir a configuração em cada dispositivo. A Tabela 15a com os agregados usados no *switch OPF1* são os registros deste dispositivo em 14c e os prefixos associados aos *switches OPF2 e OPF3* em 14d. As interfaces usadas para alcançar os outros dispositivos e o endereço IP são obtidos com o NOX.

Tabela AGG no OPF1

Destino	IFACE
0.0.0.0/5	4
192.0.0.0/3	1
128.0.0.0/2	2
0.0.0.0/1	3

(a)

Tabela AGG no OPF2

Destino	IFACE
128.0.0.0/2	4
0.0.0.0/1	2
192.0.0.0/3	1

(b)

Tabela AGG no OPF3

Destino	IFACE
192.0.0.0/6	3
192.0.0.0/3	4
128.0.0.0/2	1
0.0.0.0/1	2

(c)

Tabela 15 – FIB nos *switches* OpenFlow.

A verificação pode ser feita considerando que pacotes com destino à rede $1.10.16.0$ $/24$ chegam na interface 3 do *switch OPF3*. Este fluxo de dados é representado com setas na Figura 23. Considerando a FIB neste dispositivo (Tabela 15c), o pacote é encaminhado para a interface de saída 2 até o *switch OPF1*. O próximo caminho usa a interface 3 de acordo com a FIB do OPF1 (Tabela 15a).

4 Experimento para Validação

Neste capítulo é explicado a metodologia empregada nos experimentos para validação do algoritmo. Na Seção 4.1, são descritas as características do ambiente de rede virtualizado e as métricas usadas na verificação. Na Seção 4.2, as questões sobre a interação entre os elementos de rede e a sequência de execução dos testes são respondidas. Na Seção 4.3, os detalhes sobre a motivação, as ferramentas usadas e a forma como os dados são obtidos estão descritos, para validar um cenário de Internet com roteadores contendo a tabela global de roteamento. Para a criação da topologia usada nos experimentos foi utilizado o *script shell* do Linux listado no Apêndice D.

4.1 Método de Avaliação

Para monitorar a tabela de encaminhamento de dados e coletar as métricas usadas para verificação, foi utilizada a medição interna (*white box*) descrita na RFC-4041 (MANRAL et al., 2005). Este método consiste na escolha de um dos componentes da rede para fazer a coleta das informações, possibilitando a posterior análise. Este equipamento é denominado "dispositivo sendo testado" (DUT, *device under test*). A ideia é criar uma topologia de rede em malha semelhante a que é encontrada na Internet (TANENBAUM, 1999), possibilitando a análise da tabela de encaminhamento de dados e o comportamento do *hardware*, que são obtidos em um único roteador.

O roteador é implementado com o *software* de roteamento instalado em uma máquina virtual. A utilização do ambiente virtual possui a vantagem de permitir que topologias sejam criadas e reconfiguradas de forma mais rápida e simplificada que as instalações de *softwares* de roteamento feitas diretamente sobre sistemas operacionais hospedeiros (KALISZAN et al., 2012).

4.1.1 Virtualização de redes

Nascimento et al. (2011) propõe que o gerenciamento dos comutadores de rede utilizem as suítes de roteamento em máquinas virtuais. O uso de contêineres permite que a topologia lógica da rede criada utilizando as SRs reflita a topologia física formada pelos comutadores, o que proporciona simplicidade no gerenciamento dos protocolos de roteamento e escalabilidade do plano de controle, ao serem acrescidas MVs de acordo com o surgimento de novas topologias lógicas.

A virtualização de sistemas (BHATIA et al., 2008) possui três formas de implementação: completa, paravirtualização e baseada em contêineres.

1. No primeiro caso, cada ambiente tem total controle sobre o que vai executar, escolhendo inclusive o seu *kernel* de sistema operacional. Quando há necessidade de alocação de algum recurso de *hardware*, o pedido é recebido pelo virtualizador que o traduz e o passa para o sistema operacional da máquina. A desvantagem desta técnica é observada no consumo dos recursos computacionais. O terceiro caso explicado a seguir contorna esta situação obtendo um melhor aproveitamento da memória compartilhada.

2. O segundo caso é considerado uma variação do anterior (BHATIA et al., 2008), provendo melhores resultados através da otimização dos mecanismos de emulação do *hardware*. Para este tipo de virtualização, o sistema operacional que será instalado na MV tem que sofrer alterações para obter melhor desempenho do *hardware*. A desvantagem deste método é a necessidade de alteração de todos os sistemas operacionais convidados.

3. No terceiro caso, uma imagem do sistema operacional virtualizado é compartilhada entre os diversos ambientes, ou seja, todos os ambientes usam um mesmo sistema operacional. Não há disputa de recursos de *hardware* entre os contêineres, já que os recursos de *hardware* são divididos durante a criação dos ambientes.

Segundo Araújo et al. (2011), as técnicas de virtualização de sistemas operacionais baseadas em contêineres chegam a uma performance até duas vezes superior às outras técnicas, principalmente em cenários de serviços baseados em *web*. Além disso, a técnica de virtualização baseada em contêineres apresenta um melhor desempenho quando se demanda mais máquinas virtuais rodando simultaneamente.

4.1.2 *Softwares* de Roteamento

As suítes de roteamento (SR) são *softwares* que preenchem de forma automática as tabelas de roteamento usadas pelo *kernel* do sistema operacional para o encaminhamento dos pacotes de dados. Com esta finalidade, as SRs implementam protocolos de roteamento e disponibilizam ferramentas para a gerência da rede e interface de console para controle do ambiente. Para se ter uma ideia da importância dos *softwares* de roteamento no ambiente da Internet, basta lembrar que os IXPs (*Internet eXchange Provider*) da Europa possuem 66 *route servers* onde 37 deles usam SRs para trocar tabelas de roteamento: 15 com BIRD, 8 com Quagga, 8 com OpenBGP e 6 com outras SRs (SANGHANI, 2012). Existem dois tipos de arquitetura para as SRs (KALISZAN et al., 2012):

1. No primeiro tipo, múltiplos processos são organizados em três camadas (HANDLEY et al., 2003) que atuam sobre o *kernel*: uma camada da base de informação de roteamento (RIB, *Routing Information Base*), uma camada dos processos que implementam as funcionalidades dos protocolos de roteamento e uma camada com os processos usados para gerenciamento. A Figura 24 mostra este tipo de arquitetura, onde no topo estão os processos da interface de interação com o usuário (CLI, *Comand Line Interface*), o processo para controle remoto (SSH, *Secure shell tunneling*) e protocolo usado para gerenciamento (SNMP, *Simple Network Management Protocol*).

2. O segundo tipo de arquitetura é a monolítica. Neste caso, existe um único arquivo executável composto por módulos estaticamente ligados (TROSVIK, 2006). A cada

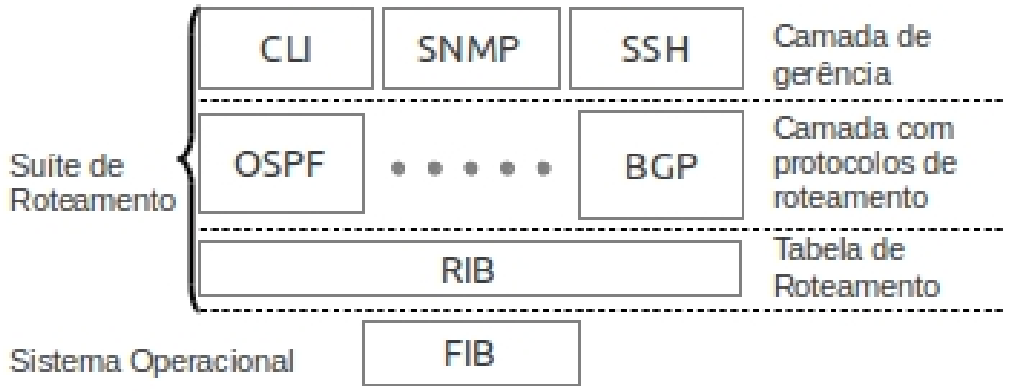


Figura 24 – Arquitetura multicamadas das suítes de roteamento.

inclusão, alteração ou deleção de módulos, um novo arquivo executável é gerado. Por exemplo, pode-se definir um módulo para cada protocolo de roteamento disponível na SR. O Quagga e o XORP são exemplos de *softwares* com múltiplas camadas, enquanto o BIRD é do tipo monolítico (FARIAS et al., 2013).

BIRD (Internet Routing Daemon)

O BIRD (ONDREJ et al., 2012) é uma suíte de roteamento muito usada em pontos de troca de tráfego (IXP, *Internet Exchange Point*) na Europa como roteador centralizador e distribuidor de rotas (*route server*) anunciadas da Internet (SANGHANI, 2012). Os módulos do BIRD oferecem serviços como gerenciamento dos recursos compartilhados, ligação com o *kernel* do sistema operacional e gerenciamento de múltiplas tabelas de roteamento. A Figura 25 mostra uma possível configuração para o BIRD no ambiente de máquinas virtuais usando contêineres.

Neste exemplo, existem dois protocolos de roteamento que alimentam a tabela primária com as rotas calculadas. Cada contêiner possui duas bases de dados do *kernel* do sistema operacional: RIB (*Routing Information Base*) e FIB (*Forwarding Information Base*). A RIB é preenchida automaticamente com o conteúdo da tabela primária. Portanto, é possível construir um ambiente de redes onde cada contêiner é um roteador e a ligação

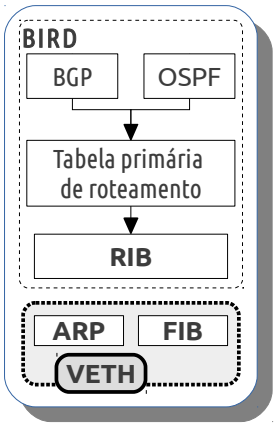


Figura 25 – Suíte de roteamento BIRD.

entre os diversos roteadores é feita através de uma interface virtual.

4.1.3 Conexão Virtual

A conectividade que permite a troca de mensagens entre as máquinas virtuais é obtida através da associação da interface de rede local (ETH, *ethernet*) e as interfaces virtuais (VETH, *Virtual Ethernet device*). Isto é feito através da configuração de uma ponte no *kernel* do Linux da máquina real (*linux bridge*) de forma obter a conexão entre as camadas *intra rede* . Cada VETH tem associado um número *ethernet* (MAC *address*) e um número IP. Desta forma, cada contêiner corresponde a um roteador e consiste em uma cópia isolada da suíte de roteamento, como ilustrado na Figura 26.

4.1.4 *Hardware* Utilizado

Para a coleta das métricas foi utilizada uma máquina com processador *Quad Core* de 64 bits, 8 GBytes de memória e sistema GNU-Linux Debian 6.0.6 como sistema operacional hospedeiro. O ambiente de testes foi criado com máquinas virtuais LXC 0.7.2, do tipo contêiner, nas quais foram instaladas a suíte de roteamento BIRD 1.3.4 (FARIAS et al., 2013).

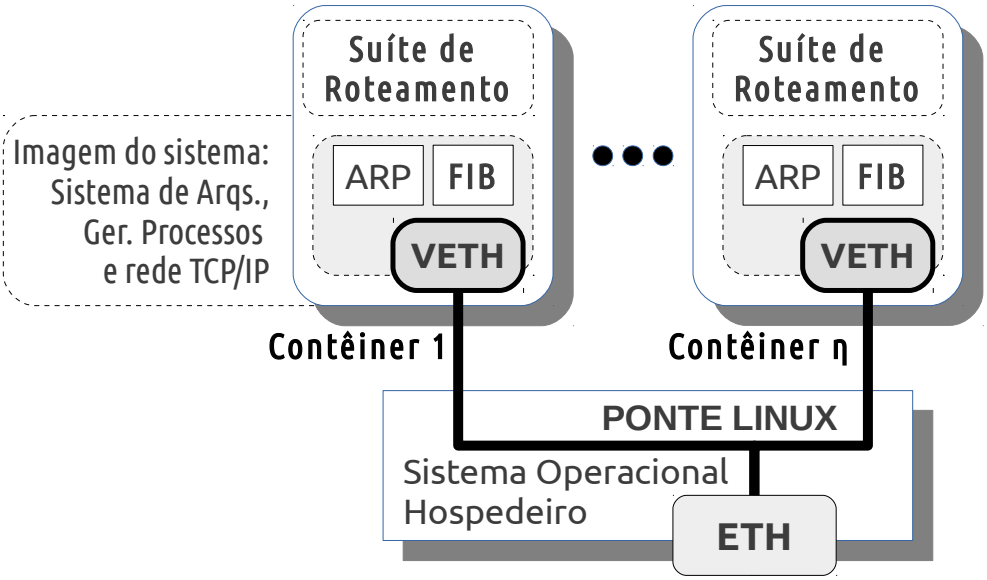


Figura 26 – Conexão entre contêineres.

4.1.5 Métricas de avaliação

Duas métricas são analisadas neste trabalho: o tamanho da tabela de encaminhamento e a latência da rede. Na primeira métrica é verificada a taxa de agregação do algoritmo proposto. Para isso, a quantidade de entradas na FIB gerada pela solução proposta é comparada com o tamanho registrado quando o BIRD faz a configuração dinâmica. Na segunda métrica, a latência da rede é medida para verificação do desempenho considerando o tamanho da FIB (TROTTER, 2001) .

Para obter uma avaliação dos resultados com relevância estatística, cada métrica é registrada em trinta execuções distintas. A garantia de que o ambiente não possui resíduos da configuração anterior é obtido com a criação de um novo, após a remoção completa do cenário anterior. Esta operação e as outras necessárias para efetivação do experimento são monitoradas por rotinas que permanecem carregadas na memória principal (*daemon*) até que o número de repetições especificado tenha sido executado. Em cada repetição, o conjunto de operações realizado está organizado em quatro fases:

FASE 1- Criação do cenário: Uma nova topologia da rede formada com suítes de rotea-

mento é criada e configurada em máquinas virtuais.

FASE 2- Carga dos aplicativos: São iniciados os *scripts* que testam a conectividade e coletam as métricas, e a suíte de roteamento é configurada e iniciada.

FASE 3- Configuração da FIB: Após ocorrer a convergência da rede, o algoritmo de redução da FIB é executado com os prefixos obtidos na suíte de roteamento. O resultado obtido é configurado na FIB (forma estática) após a suíte de roteamento ser desativada.

FASE 4- Coleta e remoção: As métricas registradas são arquivadas e o ambiente virtual é eliminado.

4.2 Implementação

As operações realizadas durante os testes seguem uma sequencia que é determinada pelo término da operação anterior ou de acordo com a utilização de uma métrica. Por isso, as rotinas operacionais são organizadas em dois *scripts*. O primeiro cria a topologia da rede, configura a máquina virtual e a suíte de roteamento, executa o algoritmo de filtragem e agregação, faz a coleta dos resultados e remove o cenário, garantindo que não existirão fragmentos de dados do experimento anterior. O segundo fica residente em memória (*daemon*) de forma a realizar o monitoramento em tempo real da carga de processamento e da latência da rede.

Para verificar a continuidade da rotas usadas até as redes de destino é necessário a alteração do conteúdo da FIB com os resultados calculados. Desta forma, é realizada a validação do algoritmo. Duas formas podem ser usadas para a adição e remoção dos prefixos: usando comandos do *kernel* para alterar os prefixos de forma estática ou com *softwares* de roteamento que alteram o conteúdo de forma dinâmica.

4.2.1 Criação do roteador virtual

Antes de iniciar os experimentos, um modelo do roteador virtual é criado em um contêiner conectado na Internet para fazer a instalação dos *softwares*, inclusive a suíte de roteamento. Esta MV é usada para a criação dos demais roteadores participantes da topologia. O molde é configurado da seguinte forma:

1. O BIRD é ajustado de forma a estar operacional com o BGP no momento da inicialização do sistema. No entanto, ele é configurado para não enviar ou receber anúncios, nem fazer alterações na configuração da tabela de encaminhamento.

2. No servidor ssh é feita instalação da chave privada usada pelo script de controle, para acessar as informações remotamente.

3. O *script* usado no monitoramento é criado e configurado para carga automática na inicialização do sistema.

4. Os arquivos de configuração de rede no Linux que incluem a atribuição do endereço IP e o *default gateway* são removidos, garantindo que o *script* de controle seja o único mecanismo que torna o dispositivo um *host* ou *gateway* na rede. Esta remoção é feita antes de um novo ciclo de testes, de forma a garantir que a nova topologia não possua roteadores configurados com os resultados obtidos em cálculos anteriores..

5. O arquivo de configuração do contêiner é removido. Com esta operação, é removido o nome do molde, as interfaces e os respectivos endereços físicos. Esta última alteração, realizada após o *shutdown* do contêiner, garante que a cópia dos arquivos não resultam na criação de um dispositivo conectado no enlace.

Na etapa de criação do contêiner, realizada pelo *script* de controle, este modelo é copiado e renomeado de acordo com a identificação do roteador na topologia. Por exemplo, na criação dos cinco roteadores usados nos experimentos (DUT, N1, N2, B1 e B2), a cópia e renomeação é realizada cinco vezes utilizando os comandos `cp` e `mv` do Linux. E no

término da rodada do experimento, o comando `rm` é usado com o parâmetro recursivo para remover completamente estes roteadores.

4.2.2 Fluxograma de funcionamento e medição

No diagrama ilustrado na Figura 27 são descritas as etapas de funcionamento das rotinas que automatizam a criação da topologia da rede e a execução dos testes. O *script* de controle tem a função de gerenciar os dispositivos e a coleta das métricas. O *daemon* de monitoramento faz o registro das métricas e interfere com a sequência de execução do experimento, pois os valores armazenados são usados pelo *script* de controle.

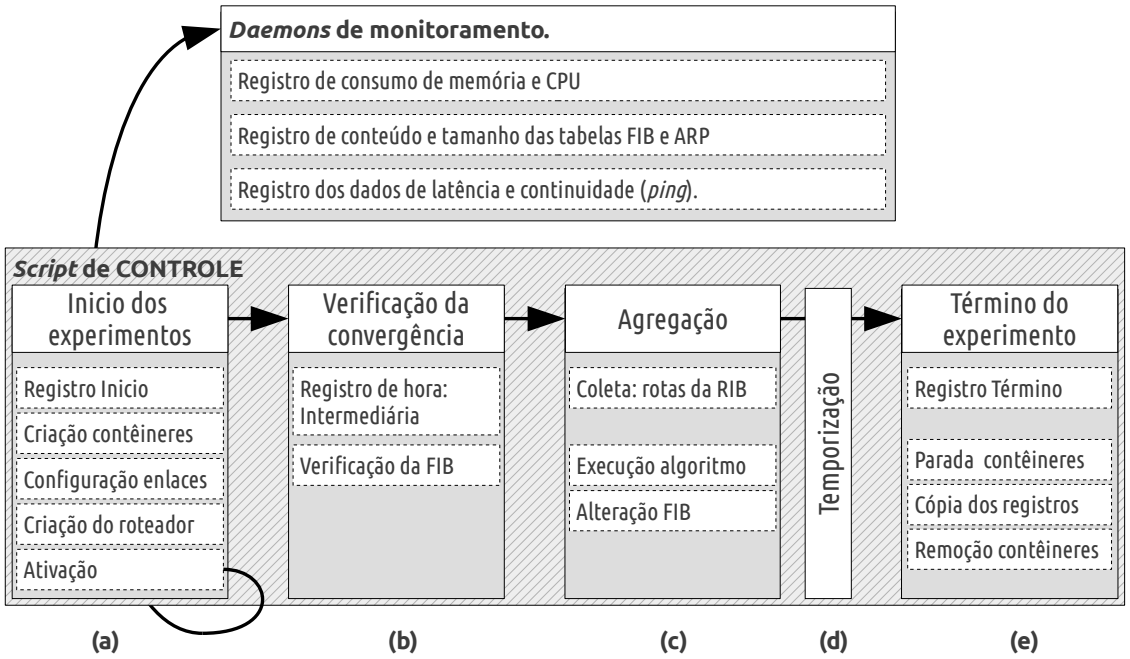


Figura 27 – Fluxograma de funcionamento e medição.

Script de controle

A) Inicio dos experimentos

1. O momento em que o *script* de controle é carregado para execução é registrado no formato de tempo universal Unix. A quantidade de segundos passados desde “01

de Janeiro de 1970" é usada para facilitar os cálculos de intervalo de tempo e as análises estatísticas que serão executados.

2. A criação dos contêineres é obtida com a cópia do molde que possui configurada a suíte de roteamento e os *softwares* usados nas medições. O diretório usado pela máquina virtual para armazenar os arquivos do contêiner são nomeados de acordo com a identificação dos dispositivos definidos na topologia da rede.

3. Na configuração do enlace, cada cópia do contêiner tem o arquivo alterado de acordo com a topologia pré-determinada. Esta configuração atribui o nome ao contêiner, as interfaces virtuais com os respectivos endereços IPs e os nomes que serão conhecidos dentro do contêiner (Figura 28) além de outros detalhes de configuração que são específicos para cada máquina virtual LXC. Isso torna o contêiner um dispositivo único conectado no enlace.

```
lxc.utsname = dut-bird

lxc.network.type = veth
lxc.network.ipv4 = 172.16.1.1/24
lxc.network.name = eth1
lxc.network.flags = up

lxc.network.type = veth
lxc.network.ipv4 = 172.16.2.1/24
lxc.network.name = eth2
lxc.network.flags = up
```

Figura 28 – Arquivo de configuração do contêiner usado pelo roteador DUT.

4. A configuração dos arquivos de rede no Linux e da suíte de roteamento transforma o contêiner em um roteador (ou *host*) específico da topologia, alterando os parâmetros de endereçamento lógico (endereço IP) e fazendo os anúncios das rota usando o BGP. A Figura 29 representa as linhas do arquivo `bird.conf` usada no roteador

DUT. No arquivo são comentados os protocolos usados para configurar as rotas diretamente conectadas, o intervalo de tempo para as alterações da FIB usando os dados da RIB e os vizinhos que são aceitos como fonte de anúncios.

```
router id 10.50.10.2;
protocol static local_DUT{              # Rotas diretamente conectadas.
   route 10.50.10.2/24 via "eth1";}
   route 10.50.20.2/24 via "eth2";}
protocol kernel {scan time 5;           # Altera a FIB com os dados da RIB
   import all}                          # verificada a cada 5 segundos
protocol bgp N1 {local as 10;           # Habilita o recebimento via BGP
   neighbor 10.50.10.1 as 10;
   import all}
```

Figura 29 – Configuração dinâmica do roteador DUT.

5. A ativação do contêiner resulta na inicialização automática dos *daemons* de monitoramento e do BIRD e caracteriza o início dos experimentos. O comando `lxc-start` é usado para iniciar o contêiner com o roteador DUT.

B) Verificação da convergência

6. Após os dispositivos serem ligados, a comparação de dois momentos nos registros da FIB possibilita a verificação do tempo médio usado para que todos os roteadores da topologia tenham calculado as rotas usadas para o encaminhamento dos pacotes.

C) Agregação

7. O conteúdo da RIB é copiado após a convergência da rede. O seletor usado nos experimentos é a rota escolhida (*best route*), marcada na RIB com um asterisco. Por isso o resultado final é o mesmo que a listagem do conteúdo da FIB, como pode ser observado na comparação dos prefixos marcados na Figura 30 e o conteúdo na Figura 31. Por isso, no *script* é usado o comando `lxc-netstat`, que obtém

o conteúdo da FIB sem a necessidade da carga do servidor SSH para o controle remoto.

```
bird> show route
10.50.10.0/24  via 10.50.10.1 on eth4 [LINKDIRETO 09:42] * (200)
        via 10.50.30.2 on eth1 [N2 10:00] (100) [AS10i]
10.50.20.0/24  via 10.50.10.1 on eth4 [DUT 09:42] * (100) [AS10i]
        via 10.50.30.2 on eth1 [N2 10:00] (100) [AS200i]
        via 10.50.40.2 on eth2 [B1 10:00] (100) [AS200i]
        via 10.50.50.2 on eth3 [B2 10:00] (100) [AS200i]
```

Figura 30 – RIB obtida no BIRD usando o comando *show*.

```
linux> ip route show
10.50.10.0/24 dev eth4  proto kernel  scope link  src 10.50.10.2
10.50.20.0/24 via 10.50.10.1 dev eth4  proto bird
```

Figura 31 – FIB obtida no Linux usando o comando *ip*.

8. O *script* de controle carrega o algoritmo de filtragem e agregação para o cálculo das novas rotas usadas. A interface de linha de comandos *BASH* do Linux permite a transferência de sessão, possibilitando que as operações do *script* de controle fiquem suspensas até que o resultado dos cálculos sejam recebidos.

   Na (Figura 32) é exemplificado como os prefixos gerados pelo algoritmo de filtragem e agregação são inseridos como rotas estáticas no BIRD (*protocol static*) e a opção dos anúncios BGP é desligada (usando o # para transformar a linha em comentário). Essa sequência altera o comportamento da obtenção das rotas de dinâmico para estático.

E) Término do experimento

10. Os contêineres são parados enviando o comando `lxc-shutdown` para todos os contêineres, o que resulta no `shutdown` dos sistemas operacionais.

```
Router id 10.50.10.2;
protocol static local_DUT{
   route 10.50.10.2/24 via "eth1";}
   route 10.50.20.2/24 via "eth2";}
protocol kernel {scan time 5;
   import all}
# protocol bgp N1 {local as 10;          # Desabilita o BGP
#     neighbor 10.50.10.1 as 10;
#     import all}
protocol static LINKDIRETO {
   Route 0.0.0.0/0   via 10.50.10.2;       # eth1
   route 10.8.0.0/13   via 10.50.20.2;     # eth2
   route 10.32.0.0/11   via 10.50.20.2;    # eth2
   route 172.16.6.0/23   via 10.50.20.2;   # eth2
   route 172.16.8.0/22   via 10.50.20.2;   # eth2
   route 192.168.6.0/23   via 10.50.20.2; # eth2
   route 192.168.8.0/22   via 10.50.20.2; # eth2
}
```

Figura 32 – Configuração estática do BIRD.

11. Os arquivos criados pelo *daemon* de monitoramento são transferidos para a base de dados central usando como identificação a hora no momento da cópia.

12. A remoção dos contêineres é obtida com o comando `lxc-destroy` usando o nome dos roteadores presentes na topologia.

Daemon de monitoramento

O *Daemon* de monitoramento é iniciado automaticamente no *boot* do sistema operacional no contêiner. As métricas são obtidas com uso de outros programas, que são carregados em intervalos de tempo de 1 segundo ou configurados para permanecerem em

execução. Os resultados são armazenados no diretório `/root`. Abaixo estão descritas as características dos quatro aplicativos usados para o monitoramento:

1. **Registro do consumo de memória e CPU:** Essas métricas são coletadas pelo *software* `sar` (*System Activity Reporter*) e registradas nos arquivos `/root/usocpu` e `/root/usomem`. Este aplicativo tem que ser acrescentadas, pois não faz parte do pacote que acompanha o Linux.

2. **Registro do tamanho das tabelas FIB e ARP:** O comando `ip` pertence ao pacote `iproute2` instalado por padrão com o Linux. Esta ferramenta possibilita a coleta das métricas e a alteração da tabela de roteamento, dispositivos de rede, políticas de roteamento, entre outras. O *daemon* de monitoramento executa este comando em intervalos regulares, obtendo o conteúdo da FIB (`ip route show`) e da tabela ARP (`ip neigh show`). Essas informações são registradas nos arquivos `/root/iprouteshow` e `/root /ipneighshow`.

3. **Registro da latência e continuidade usando o *ping*.** O comando `ping` apresenta o tempo que leva para as requisições de resposta ICMP *Echo Request* serem respondidas com ICMP *Echo Replay*. Os resultados são usados para verificar se as interfaces dos roteadores são alcançáveis, possibilitando o registro do tempo necessário para os pacotes irem e voltarem.

4.2.3 Cenário de Testes

Três ASN são usados, onde o roteador DUT (*device under test*) é o único no AS10, os roteadores N1 e B1 estão no AS100 e os roteadores N2 e B2 no AS200. os função do DUT no cenário é coletar as métricas para análise (MANRAL et al., 2005), por isso ele recebe os anúncios BGP, envia pacotes ICMP e mantém registros de uso da memória e CPU. O núcleo é formado pelos roteadores DUT, N1 e N2 e as bordas pelos roteadores B1 e B2 que dão acesso às redes clientes. Trinta e duas rotas são anunciadas por ISPs subordinados e estão caracterizadas como nuvens conectadas às bordas. A existência de

caminhos redundantes nos clientes foi obtida com a topologia em malha totalmente ligada entre N1, N2, B1 e B2. A Figura 33 representa essa topologia.

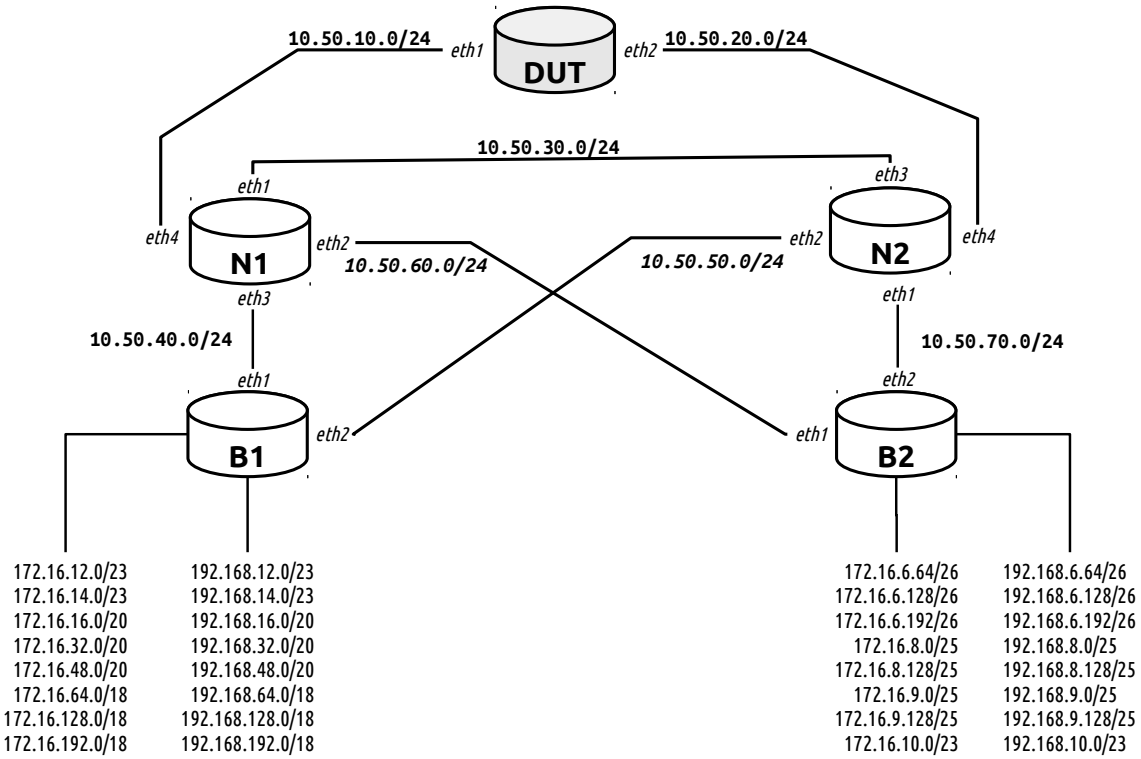


Figura 33 – Cenário usado para os testes.

4.3 Experimentos para avaliação

Nesta seção são apresentados experimentos executados para avaliação da proposta através da simulação de uma topologia tipicamente encontrada no núcleo da Internet. Nesta região os roteadores são configurados sem a rota padrão (DFZ, Default Free Zone), por isso a tabela de rotas tem o prefixo de todos os endereços que podem ser alcançados. Utilizou-se suítes de roteamento BIRD criadas com máquinas virtuais LXC. A eficiência da proposta foi calculada com os resultados obtidos no monitoramento do tamanho da FIB e da latência da rede, além da verificação da alcançabilidade das redes originais.

4.3.1 Redução da FIB

A RIB obtida no DUT apresenta 68 rotas para alcançar os prefixos anunciados, incluindo os caminhos alternativos. Destas, 44 prefixos de destino e os respectivos endereços de próximo salto são passados para a FIB, conforme a solução tradicional implementada no BIRD, que é observada na Figura 34. Isso permite observar a simplificação usual na FIB, pois 35,2% das rotas não são registradas. A FIB do BIRD foi coletada através do comando *ip route show*.

```
~\# ip route show
10.10.10.0/24 dev eth0 proto kernel scope link src 10.10.10.1
10.10.30.0/24  via 10.50.10.2 dev eth1 proto bird
10.10.60.0/24  via 10.50.20.2 dev eth2 proto bird
10.10.80.0/24  via 10.50.10.2 dev eth1 proto bird
10.10.100.0/24  via 10.50.20.2 dev eth2 proto bird
10.50.10.0/24 dev eth1 proto kernel scope link src 10.50.10.1
10.50.20.0/24 dev eth2 proto kernel scope link src 10.50.20.1
10.50.30.0/24  via 10.50.10.2 dev eth1 proto bird
10.50.40.0/24  via 10.50.10.2 dev eth1 proto bird
10.50.50.0/24  via 10.50.10.2 dev eth1 proto bird
10.50.60.0/24  via 10.50.10.2 dev eth1 proto bird
10.50.70.0/24  via 10.50.20.2 dev eth2 proto bird
172.16.6.64/26  via 10.50.20.2 dev eth2 proto bird
172.16.6.128/26  via 10.50.20.2 dev eth2 proto bird
172.16.6.192/26  via 10.50.20.2 dev eth2 proto bird
172.16.8.0/25  via 10.50.20.2 dev eth2 proto bird
172.16.8.128/25 via 10.50.20.2 dev eth2 proto bird
172.16.9.0/25 via 10.50.20.2 dev eth2 proto bird
172.16.9.128/25 via 10.50.20.2 dev eth2 proto bird
172.16.10.0/23 via 10.50.20.2 dev eth2 proto bird
172.16.12.0/23 via 10.50.10.2 dev eth1 proto bird
172.16.14.0/23 via 10.50.10.2 dev eth1 proto bird
172.16.16.0/20 via 10.50.10.2 dev eth1 proto bird
172.16.32.0/20 via 10.50.10.2 dev eth1 proto bird
```

```
172.16.48.0/20 via 10.50.10.2 dev eth1 proto bird
172.16.64.0/18 via 10.50.10.2 dev eth1 proto bird
172.16.128.0/18 via 10.50.10.2 dev eth1 proto bird
172.16.192.0/18 via 10.50.10.2 dev eth1 proto bird
192.168.6.64/26 via 10.50.20.2 dev eth2 proto bird
192.168.6.128/26 via 10.50.20.2 dev eth2 proto bird
192.168.6.192/26 via 10.50.20.2 dev eth2 proto bird
192.168.8.0/25 via 10.50.20.2 dev eth2 proto bird
192.168.8.128/25 via 10.50.20.2 dev eth2 proto bird
192.168.9.0/25 via 10.50.20.2 dev eth2 proto bird
192.168.9.128/25 via 10.50.20.2 dev eth2 proto bird
192.168.10.0/23 via 10.50.20.2 dev eth2 proto bird
192.168.12.0/23 via 10.50.10.2 dev eth1 proto bird
192.168.14.0/23 via 10.50.10.2 dev eth1 proto bird
192.168.16.0/20 via 10.50.10.2 dev eth1 proto bird
192.168.32.0/20 via 10.50.10.2 dev eth1 proto bird
192.168.48.0/20 via 10.50.10.2 dev eth1 proto bird
192.168.64.0/18 via 10.50.10.2 dev eth1 proto bird
192.168.128.0/18 via 10.50.10.2 dev eth1 proto bird
192.168.192.0/18 via 10.50.10.2 dev eth1 proto bird
```

Figura 34 – Solução do BIRD para a FIB no DUT.

Com a utilização do algoritmo proposto, a FIB obtida apresenta um total de 15 prefixos de rede, o que equivale a uma redução de 65,91% com relação à FIB obtida com o esquema tradicional. Este resultado é listado na Figura 35.

```
~# ip route show
10.10.10.0/24 dev eth0 proto kernel scope link src 10.10.10.1
10.10.30.0/24 via 10.50.10.2 dev eth1 proto bird
10.10.60.0/24 via 10.50.20.2 dev eth2 proto bird
10.10.80.0/24 via 10.50.10.2 dev eth1 proto bird
10.10.100.0/24 via 10.50.20.2 dev eth2 proto bird
10.50.10.0/24 dev eth1 proto kernel scope link src 10.50.10.1
```

```
10.50.20.0/24 dev eth2 proto kernel scope link src 10.50.20.1
10.50.70.0/24 via 10.50.20.2 dev eth2 proto bird
172.16.12.0/22 via 10.50.10.2 dev eth1 proto bird
192.168.12.0/22 via 10.50.10.2 dev eth1 proto bird
172.16.0.0/20 via 10.50.20.2 dev eth2 proto bird
192.168.0.0/20 via 10.50.20.2 dev eth2 proto bird
10.50.0.0/17  via 10.50.10.2 dev eth1 proto bird
172.16.0.0/16 via 10.50.10.2 dev eth1 proto bird
192.168.0.0/16  via 10.50.10.2 dev eth1 proto bird
```

Figura 35 – Solução do algoritmo para a FIB.

4.3.2 Latência no Cenário Virtual

A verificação da influência da redução da FIB no tempo de ida e volta de um pacote foi obtida com o protocolo ICMP (*Internet Control Message Protocol*). Para isso, foram usados pacotes com tamanho de 900 bytes em rajadas sem confirmação, com intervalos de tempo nulo, de forma a repetir o tráfego característico neste tipo de rede (CÁCERES et al., 1991) . O comando `ping -s900 -i0 B1` foi usado para fazer o envio dos pacotes ICMP, onde B1 é o roteador de borda usado para acesso ao núcleo da rede.

A Tabela 16 relaciona as médias dos tempos mínimo, médio e máximo de ida e volta dos pacotes (RTT - *Round Trip Time*) enviados para as SRs B1 e B2 a partir do DUT, nas trinta rodadas de teste. Em média, é observado uma melhora de 31% na latência da rede para este ambiente controlado.

	min.	**méd.**	**máx.**
Sem redução	0,077	0,199	0,759
Com redução	0,062	0,137	0,303

Tabela 16 – Tempo de ida e volta em milissegundos.

Essa redução da latência ocorre porque o tempo de busca na tabela de encaminhamento é menor, consequentemente a criação do quadro de envio é realizado mais

rapidamente.

4.3.3 Verificação da continuidade das rotas

Esta verificação tem que ser realizada para todas as quarenta e quatro rotas da FIB original, porém usando a arquitetura com FIB reduzida. Para isso, os prefixos das tabelas são correlacionados de forma que a interface de saída seja verificada. As Figuras 36 e 37 permitem verificar que as rotas usada nos dois casos são iguais.

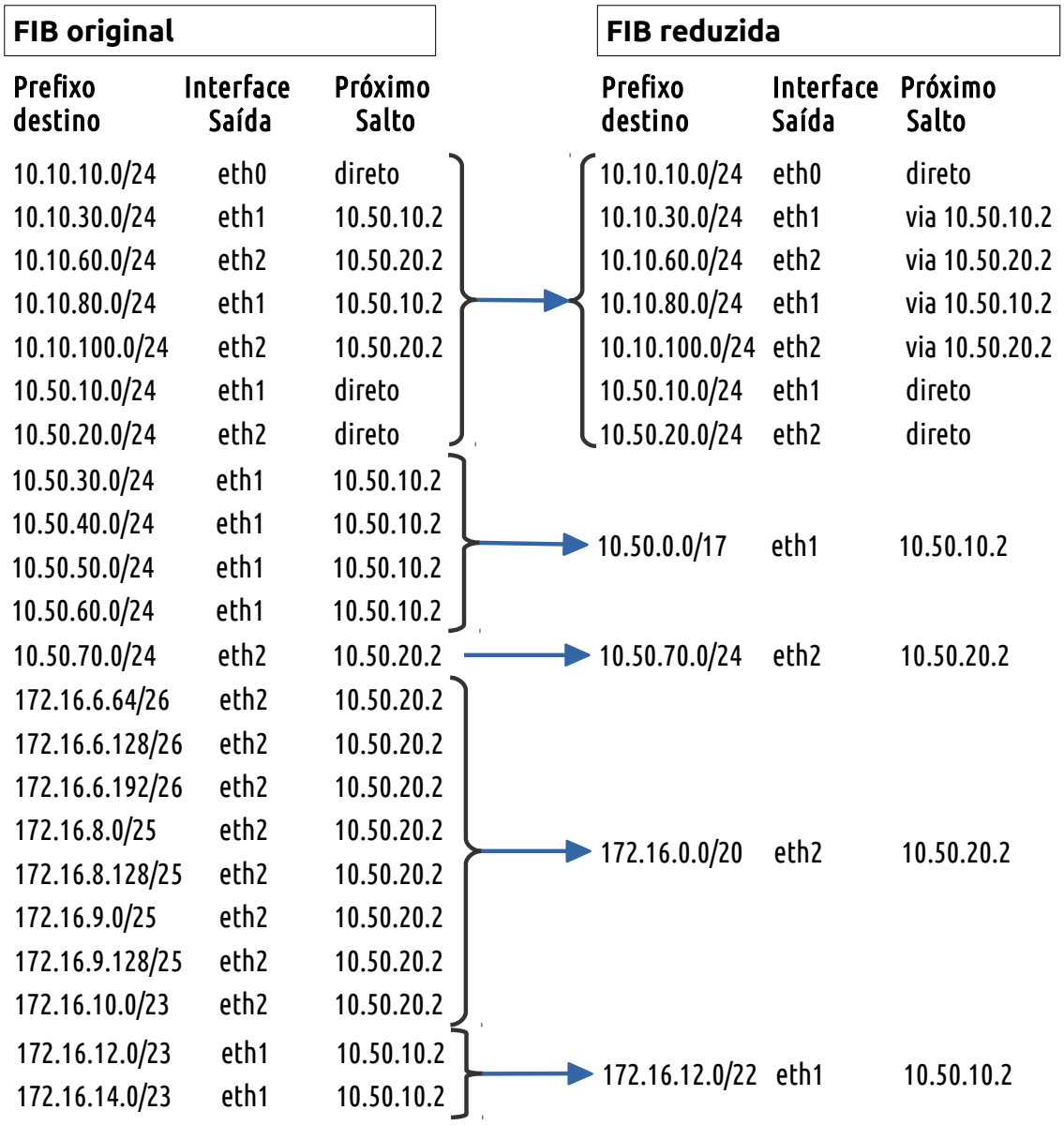


Figura 36 – Correlação FIB original X reduzida.

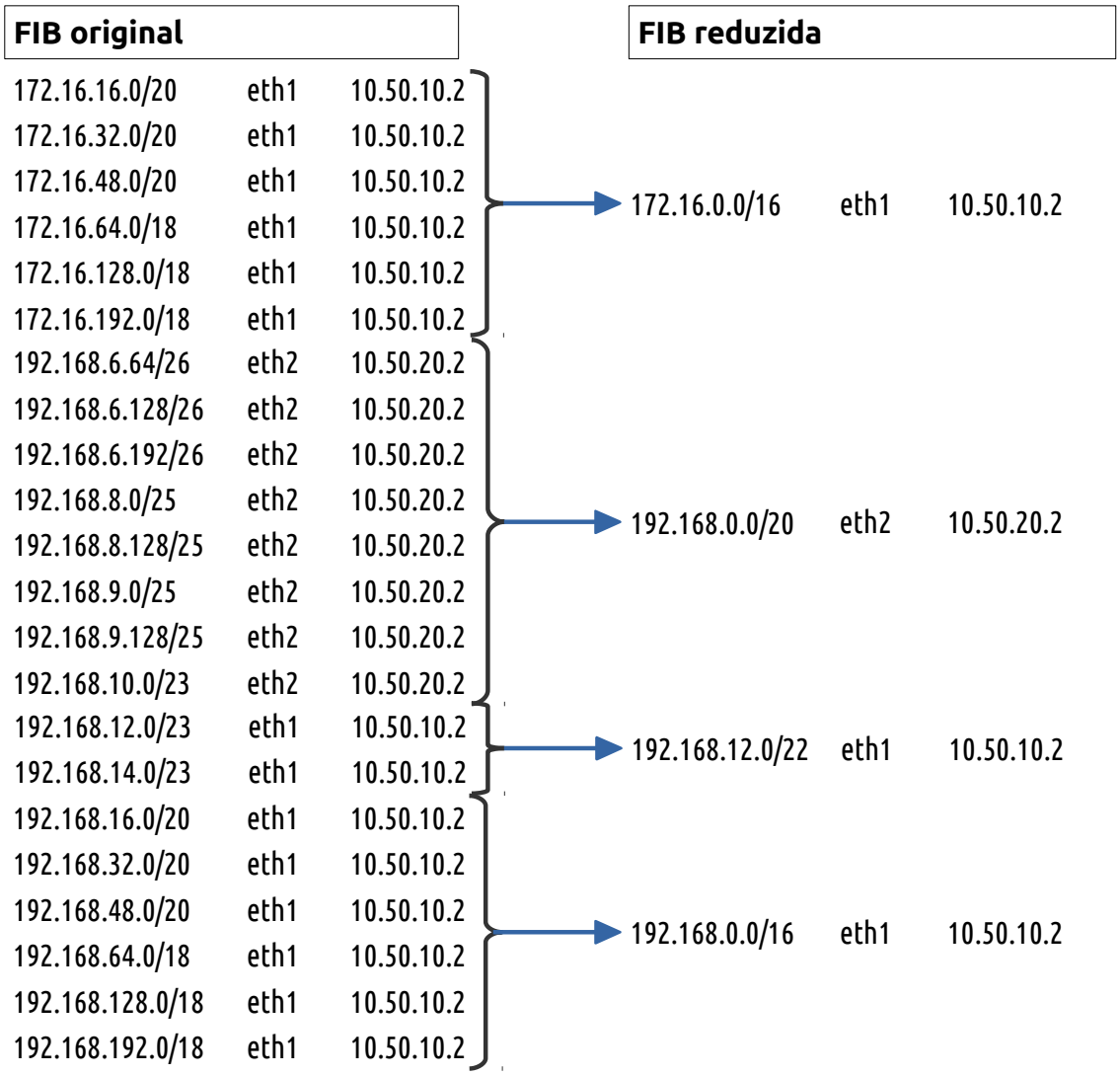


Figura 37 – Correlação FIB original X reduzida (*continuação*).

Na Figura 38 é apresentado o resultado obtido com o comando `traceroute` para um dos prefixos. O mesmo pode ser verificado com os outros 43 prefixos da FIB original. A adição de dois *hosts* no cenário foi necessária para obter o traçado fim-a-fim, uma vez que se desejava verificar todos os roteadores intermediários. Por isso, a origem do comando é um *host* conectado no roteador *B1* e o destino está conectado no roteador *B2*. Outro detalhe da configuração é que a solução da FIB reduzida foi instalada apenas no DUT, os demais roteadores foram configurados para usar a FIB original.

```
~# traceroute 192.168.10.2
traceroute to 192.168.10.2(192.168.10.2), 30 hops max, 60 byte packets
1 172.16.12.2 (172.16.12.2)  0,137 ms  0,303 ms  0,189 ms
2 10.50.40.1 (10.50.40.1)  0,137 ms  0,303 ms  0,189 ms
3 10.50.10.1 (10.50.10.1)  0,137 ms  0,303 ms  0,189 ms
4 10.50.20.2 (10.50.20.2)  0,137 ms  0,303 ms  0,189 ms
5 10.50.70.2 (10.50.70.2)  0,137 ms  0,303 ms  0,189 ms
6 192.168.10.2 (192.168.10.2)  0,137 ms  0,303 ms  0,189 ms
```

Figura 38 – Verificação da continuidade das rotas.

5 Avaliação em um Cenário Real

Nesta seção são apresentados os resultados dos experimentos executados para validação da proposta, em um cenário com suítes de roteamento, e para avaliação da taxa de agregação obtida na tabela de roteamento global. Na Seção 5.1, a tabela de roteamento global é analisada usando as conclusões obtidas no relatório CIDR (HUSTON et al., 2014). Na Seção 5.2 é explicada como a RIB na Internet foi obtida através dos servidores do projeto Route Views. Na Seção 5.3 é apresentado o estudo de caso e os resultados das métricas obtidas com a aplicação, no algoritmo proposto, dos prefixos reais usados na Internet.

5.1 Implementação

Os anúncios BGP na Internet são monitorados de forma a criar análises estatísticas e relatórios que podem ser obtidos no site <http://www.cidr-report.org> apoiado pelo APNIC, que é o órgão responsável pelos registros de endereçamento da Internet na região da Ásia e Pacífico (IANA, 2014). Entre as informações que podem ser obtidas, a projeção de redução da tabela de roteamento global é obtida com o cálculo da sumarização de rotas dos prefixos anunciados pelos provedores de serviço. Este documento é denominado *Relatório de Agregação CIDR* e pode ser obtido em <http://www.cidr-report.org/as2.0/aggr.html>.

A agregação indicada é calculada apenas quando existe a correspondência exata do caminho entre os sistemas autônomos (HUSTON et al., 2014). A Tabela 17 é um resumo do relatório com apenas os dez primeiros provedores brasileiros. Os sistemas autônomos que anunciam uma quantidade maior de prefixos do que necessário são listados primeiramente de forma a chamar a atenção para a contribuição, caso façam anúncios sumarizados. Este relatório foi gerado em 22 de junho de 2014 com base nos anúncios feitos nesta data.

#	AS	*Name*	*Current*	*Wthdw*	*Aggte*	*Annce*	*Redctn*	*%*
Routing Table			*506085*	*268713*	*45846*	*283218*	*222867*	*44,04%*
1	28573	NET Serviços	3765	3692	92	165	3600	95,62%
5	18881	Global Village Telecom	2047	2015	7	39	2008	98,09%
20	7738	Telemar Norte Leste	979	831	64	212	767	78,35%
27	26615	Tim Celular	861	794	46	113	748	86,88%
99	26599	Telefônica Brasil	369	347	15	37	332	89,97%
255	13591	Brasil Telecom	211	155	21	77	134	63,51%
397	28281	VCB provedor de acesso	214	118	34	130	84	39,25%
427	19089	DH&C Outsourccing	120	98	19	41	79	65,83%
571	28303	Rede OptiTel	77	64	5	18	59	76,62%
598	28661	Hotlink Internet	63	62	5	6	57	90,48%

Tabela 17 – Relatório com a proposta de agregação criado pelo CIDR-Report.org.

O campo # foi acrescentado para indicar a posição das empresas no relatório original. Na segunda linha estão os totais de cada campo que equivale aos valores da tabela de roteamento global. Os demais registros estão organizados da seguinte forma nos campos:

***AS* e *Name*:** É a identificação numérica e a razão social do sistema autônomo.

***Current*:** É a quantidade atual de anúncios feitos por este AS. Este número é obtido na contagem dos prefixos associados ao mesmo AS.

***Wthdw*:** É o número de prefixos substituídos pelos agregados em *Aggte*. Este valor é calculado pelo algoritmo que gerou o relatório.

***Aggte*:** É o número de prefixos agregados que substituem a quantidade em *Wthdw*. Valor calculado pelo algoritmo que gerou o relatório.

***Annce*:** É a quantidade necessária de anúncios usando a agregação. Este valor é o total de prefixos usados contabilizando a substituição dos blocos de prefixos pelos agregados ($Annce = (Current - Wthdw) + Aggte$).

***Redctn*:** É a quantidade total de prefixos que deixam de ser anunciados. O valor é obtido com a subtração da quantidade atual de anúncios da quantidade necessária ($Redctn = Current - Annce$).

%: É a taxa de redução dos anúncios. O valor é a relação entre a quantidade de prefixos removidos com a agregação e a quantidade de anúncios sem agregação ($\% = \frac{Redctn}{Current}$).

5.2 Obtenção da RIB no RouteViews

Os pesquisadores podem acessar a tabela de roteamento global usando roteadores da Internet, configurados com o protocolo BGP para receber anúncios de outros roteadores. Estes equipamentos funcionam como "telescópios" na Internet (*Looking Glasses*) e são configurados para retransmitir os anúncios para uma base de dados central. Por exemplo, em São Paulo roteadores do PTT Metro e do NIC.br fazem anúncios para o (*Looking Glass*) `200.160.6.217`, que é implementado com a suíte de roteamento Quagga.

O centro de tecnologia avançada de redes na Universidade do Oregon, Estados Unidos, possibilita o acesso à tabela de roteamento global através do projeto Route Views (MEYER, 2014). Roteadores em diversas regiões enviam as RIBs para a base de dados central periodicamente, o que possibilita o estudo cronológico da conectividade na Internet.

Estes arquivos usam o padrão recomendado pelo IETF para exportação das informações de roteamento (MRT, *Multi-Threaded Routing Toolkit*) (BLUNK et al., 2011). Também é possível acessar a tabela de roteamento em tempo real usando uma conexão remota para um dos *Looking Glasses*.

Em <http://archive.routeviews.org/route-views.saopaulo/bgpdata/> estão arquivados os registros da RIB da região brasileira desde março de 2003. Para obter os dados em tempo real é necessário acessar o respectivo servidor de *Looking Glass* usando o comando `telnet route-views.saopaulo.routeviews.org`. A Figura 39 representa o diálogo inicial de acesso.

A Figura 40 representa a tabela de roteamento global obtida em 03 de julho de

```
~$ telnet route-views.saopaulo.routeviews.org
Trying 200.160.6.217...
Connected to route-views.saopaulo.routeviews.org.
Escape character is ']'.

Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

route-views.saopaulo.routeviews.org>
```

Figura 39 – Diálogo inicial de acesso no *Looking Glass*.

2014. Para as explicações que se seguem estão sendo listados apenas dois prefixos entre os 517.259 registrados.

De acordo com o *código de situação*, um prefixo válido é representado por "$*$" e o *melhor caminho* é indicado por "$>$". Nesta listagem, a rede de destino (*Network*) `0.0.0.0` (anúncio de *rota default*) tem como melhor caminho o próximo salto (*Next Hop*) `187.16.218.57` enquanto a rede `1.0.0.0/24` usa o `187.16.216.55` como próximo salto.

Embora se trate de um tabela global, alguns ASes aceitam anúncios de *rota default* para servir de *backup* em caso de falhas.

5.3 Estudo de caso

Para avaliar o resultado obtido com a agregação usando o algoritmo proposto é necessário obter a tabela de roteamento global com pelo menos o campo de próximo salto para ser usado nos módulos de seleção, agrupamento e filtragem da proposta. Por isso, o roteador Quagga usado como *Looking Glass* do projeto Route Views foi acessado remotamente.

Um total de 517.259 prefixos foram obtidos em 03 de julho de 2014. Este valor é

```
route-views.saopaulo.routeviews.org> show ip bgp
BGP table version is 0, local router ID is 187.16.216.223
Status codes: s suppressed, d damped, h history, * valid, > best,
          i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete
Legend: M - Metric, L - LocPrf, W - Weight

  Network    Next Hop      M L W Path
*  0.0.0.0    187.16.218.21     0 52888 1251 i
*           187.16.216.20     0 28571 1251 i
*>          187.16.218.57     0 28294 i
*  1.0.0.0/24  187.16.218.121    0 52720 52873 15169 i
*           187.16.218.21     0 52888 1251  1516 i
*>          187.16.216.55     0 28329 15169 i
*           187.16.218.21     0 52888 1916 6939 7545 56203 i
...
Total number of prefixes 517259
```

Figura 40 – Tabela de roteamento global obtido em 03/Julho/2014.

superior aos 506.085 listados no relatório criado em 22 de junho (Tabela 17) porque todas as rotas são listadas no terminal do roteador, enquanto o relatório CIDR lista apenas as correspondências exatas entre os caminhos (HUSTON et al., 2014).

Os resultados parciais obtidos nas etapas do algoritmo são listados nas tabelas que se seguem. Os prefixos são ordenados e selecionados usando o critério da melhor rota assinalada na RIB. O resultado da ordenação pode ser verificado na Tabela 18a, onde são mostrados os vetores resultantes das operações de ordenação e seleção das rotas assinaladas como as escolhidas na RIB. A separação em grupos é representada na Tabela 18b e a etapa de filtragem na Tabela 19.

Os 201.687 prefixos foram obtidos como resultado da etapa de redução e representa 61,01% do total original. O relatório de agregação CIDR (Tabela 17) é usado para

AGG[0.0.0.0] =	187.16.218.57
AGG[1.0.0.0/24] =	187.16.216.55
AGG[1.0.4.0/24] =	187.16.216.4
AGG[1.0.5.0/24] =	187.16.216.4
AGG[1.0.6.0/24] =	187.16.216.4
AGG[1.0.7.0/24] =	187.16.216.4
AGG[1.0.20.0/23] =	187.16.216.232
...	

(a)

AGG[0.0.0.0] =	187.16.218.57
AGG[1.0.0.0/24] =	187.16.216.55
AGG[1.0.4.0/24] =	187.16.216.4
AGG[1.0.5.0/24] =	187.16.216.4
AGG[1.0.6.0/24] =	187.16.216.4
AGG[1.0.7.0/24] =	187.16.216.4
AGG[1.0.20.0/23] =	187.16.216.232
...	

(b)

Tabela 18 – (a) Extração e (b) agrupamento da tabela de roteamento global.

GRP[0]=187.16.218.57	IPx[0]=0.0.0.0	IPy[0]=0.0.0.0
GRP[1]=187.16.216.55	IPx[1]=1.0.0.0/24	IPy[1]=1.0.0.0/24
GRP[2]=187.16.216.4	IPx[2]=1.0.4.0/24	IPy[2]=1.0.7.0/24
GRP[3]=187.16.216.232	IPx[3]=1.0.20.0/23	IPy[3]=1.0.38.0/24
GRP[4]=187.16.216.4	IPx[4]=1.0.39.0/24	IPy[4]=1.0.40.0/24
GRP[5]=187.16.216.232	IPx[5]=1.0.50.0/24	IPy[5]=1.0.50.0/24
...	...	...

Tabela 19 – Filtragem da tabela de roteamento global.

comparação. Na Tabela 20, o campo *Atual* indica o número de prefixos anunciados sem agregação, o campo *Anunciados* é a quantidade com agregação e o campo *Removidos* é o número de prefixos que deixam de ser anunciados.

Técnica	***Atual***	***Anunciados***	***Removidos***	***%***
Seleção e Filtragem	517259	201687	315572	61,01%
CIDR *Report*	506085	283218	222867	44,04%

Tabela 20 – Comparação da técnica proposta com o Relatório CIDR.

Com o valor obtido na taxa resultante, é possível dizer que a proposta é (72,13%) mais eficiente do que o mecanismo de agregação usado atualmente na Internet.

6 Conclusão e Trabalhos Futuros

O aumento da FIB é um fator tecnológico que causa impacto no desempenho dos roteadores e que tem se intensificado com o emprego de virtualização e SDN. Neste contexto, este trabalho propõe um algoritmo para redução da FIB sob a ótica do enlace de saída para a rede. Este algoritmo usa uma abordagem na qual são criados agrupamentos de prefixos de destino cujos endereços de próximo salto estejam conectados através de uma mesma interface de rede. Procura-se então obter o menor número possível de agrupamentos que sejam associados a prefixos que exclusivamente representem faixas de rede distintas que possuam uma mesma interface de saída. Neste contexto nosso trabalho é uma técnica híbrida de filtragem e agregação.

A proposta explora a característica hierárquica de endereçamento dos provedores de serviço e a topologia da rede para obtenção, de forma mais simples, dos prefixos agregados, além de possibilitar o uso concomitante com outras técnicas, como o CIDR e agregação com algoritmos de busca em árvores. Outra vantagem na forma que foi implementado é a possibilidade da substituição de um dos módulos por outras soluções tecnológicas, como o algoritmo *Mergesort*, usado no módulo de ordenação, que pode ser substituído caso existam restrições na carga de processamento utilizada.

O algoritmo implementado é composto por cinco módulos. Os dois primeiros módulos (obtenção e ordenação), formam a etapa de extração dos dados cujo resultado final é a tabela com os prefixos selecionados na RIB. O terceiro, quarto e quinto módulos (agrupar, filtrar e agregar), formam a etapa de redução que foi implementada com a ideia de filtragem e agregação dos grupos selecionados de acordo com um critério administrativo. O resultado desta segunda etapa é a tabela com os prefixos reduzidos que são usados na FIB. O quinto e último módulo (gravação) instala na FIB o conteúdo da segunda tabela com os prefixos reduzidos.

Para avaliar o funcionamento da solução proposta e a consequente melhora de desempenho pela redução da FIB, utilizou-se suítes de roteamento BIRD executadas em ambiente virtualizado e conectadas numa topologia que emula o núcleo da Internet. Métricas associadas com a redução da FIB e a consequente redução da latência foram analisadas em experimentos realizados num cenário DFZ, ou seja, sem rota padrão, e para validar o funcionamento do sistema proposto com a tabela de roteamento global, a base de dados do projeto Route Views (MEYER, 2014) foi consultada, constatando-se a melhoria obtida com o algoritmo proposto em ambos os testes.

6.1 Contribuições da Pesquisa

- Um ponto de vista que seleciona os prefixos de rede considerando os outros registros disponíveis no protocolo de roteamento (BGP ou OSPF), além da tradicional métrica *próximo salto* usado nas pesquisas anteriores de agregação.
- Solução de redução da FIB sob a ótica da interface e da topologia da rede semelhante à proposta de agregação usando endereços *MAC* e *virtual next hop* em dispositivos OpenFlow (BRAUN; MENTH, 2014).
- Implementação da técnica de filtragem em uma única FIB, que vai além das propostas que filtram a RIB para configurar várias FIBs.
- Testes de implementação em ambiente virtual com elementos empregados nas redes reais, com foco no escalonamento de usuários na Internet.
- Emprego simultâneo com outros mecanismos usados para redução da FIB, como o *Simple Virtual Aggregation* (Seção 2.3) e o CIDR *Classless Inter-Domain Routing* (Subseção 2.1.3) e a aderência com a tecnologia usada nas redes definidas por software (SÁNCHEZ, 2013).

- Registro de métricas utilizadas em cenário criado em laboratório com dados arquivados em base de dados pública, para serem comparadas com outras técnicas a serem obtidas em pesquisas futuras.

6.2 Publicações

No decorrer do trabalho dois artigos foram publicadas em *Workshops* da Sociedade Brasileira de Computação:

XXXIII *Workshop* em Desempenho de Sistemas Computacionais e de Comunicação

Um Estudo Comparativo de Softwares de Roteamento para Uso em Redes Definidas por Software. 2013. Luiz Fernando T. de Farias, *D. Sc.* Morganna Carmem Diniz, *D. Sc.* Sidney Cunha de Lucena e *M. Sc.* Carlos Nilton Araújo Corrêa.

XXXIV *Workshop* em Desempenho de Sistemas Computacionais e de Comunicação

Uma abordagem para redução da tabela de encaminhamento sob a ótica da interface de saída dos pacotes. 2014. Luiz Fernando T. de Farias, *D. Sc.* Morganna Carmem Diniz e *D. Sc.* Sidney Cunha de Lucena.

6.3 Trabalhos Futuros

Dentre as possibilidades que já foram retratadas no texto, as apresentadas nesta seção foram consideradas durante as pesquisas:

- Adoção do seletor de prioridades usados em dispositivos OpenFlow, de forma a usar na filtragem o *wildcard-capable match* ao invés do *Longest-Prefix Matching*, otimizando a proposta em SDN.
- Incorporação da *heurística Espresso*, proposto em Braun e Menth (2014) para dispositivos OpenFlow e do *prefix TRIE*, proposto por Uzmi et al. (2011) no ORTC de forma a verificar as vantagens e melhorias no emprego dessas propostas.

- Emprego do *software* OpenVSwitch (dispositivo programável) para obter cenários puramente SDN ou híbridos. Dessa forma, comparar as soluções em dispositivos OpenFlow daquelas criadas em arquitetura legada contribuindo com resultados "normalizados" em um mesmo cenário "realista", como proposto em Braun e Menth (2014).

- Acréscimo do suporte aos endereços lógicos NDN (*Named Data Networking*) e IPv6.

6.4 Considerações Finais

A proposta apresentada unifica as ideias de técnicas de agregação e filtragem, tendo como referência a interface do roteador conectada no enlace. Por isso, os módulos podem ser substituídos por outros, seguindo um princípio equivalente ao da arquitetura em camadas. Cada módulo gera um resultado que é usado pelo próximo, e pode ser alterado para obter mais eficiência.

Além de obter o resultado esperado, constatado na análise estatística, a aplicabilidade nas redes definidas por *software* foram discutidas com a ênfase de que os dispositivos nesta arquitetura podem ser gerenciados como um grupo de interface. E as configurações em comum podem ser mescladas sem que o resultado final seja alterado.

A aplicabilidade do trabalho na arquitetura das redes definidas por software consiste principalmente na redução dos tamanhos das tabelas de encaminhamento IP sem impactos colaterais. Desta forma, as memórias dos equipamentos no plano de dados da rede podem ser economizadas, especialmente em redes OpenFlow. Além disso, nos roteadores IP a proposta é importante para estender a vida útil dos equipamentos, que se torna mais relevante com o uso de prefixos de rede IP versão 6.

Na Subseção 3.4.3 foi apresentada uma ideia em que a proposta unifica o S-VA e o protocolo de seleção da FIB (MOHAMED et al., 2012) como uma forma de economizar

memória e processamento do plano de controle, agregando a configuração que existe em comum. Esta classificação remete a trabalhos relacionados com a otimização do tráfego.

A validade do tema e da contribuição desta dissertação é observada na constatação de que a proposta vai de encontro às pesquisas mais recentes, para descobrir novas formas de melhorar o desempenho dos roteadores na Internet ao mesmo tempo que garantindo o ingresso de novos participantes na rede, como por exemplo o *Software Defined Internet Exchange* (GUPTA et al., 2014).

Referências

ARAÚJO, C. N.; LUCENA, S. C.; CORRÊA; ROTHENBERG, C.; SALVADOR, M. Uma Avaliação Experimental de Soluções de Virtualização para o Plano de Controle de Roteamento de Redes Virtuais. In: *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. [S.l.]: SBC Sociedade Brasileira de Computação, 2011.

ARAÚJO, C. N.; LUCENA, S. C.; CORRÊA; ROTHENBERG, C.; SALVADOR, M. Uma plataforma de roteamento como serviço baseada em redes definidas por software. In: *XVII Workshop de Gerência e Operação de Redes e Serviços*. [S.l.]: SBC Sociedade Brasileira de Computação, 2012.

BAKER, F. *RFC1812*: Requirements for IP Version 4 Routers. RFC Editor, 1982. Category: Standards Track. (Network Working Group. Request for Comments, 1812). Disponível em: <http://tools.ietf.org/html/rfc1812>. Acesso em: março/2014.

BALLANI, H.; FRANCIS, P.; CAO, T.; WANG, J. Making Routers Last Longer with ViAggre. In: *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2009. (NSDI'09), p. 453–466. Disponível em: <http://dl.acm.org/citation.cfm?id=1558977.1559008>.

BERKOWITZ, H.; DAVIES, E. B.; HARES, S.; KRISHNASWAMY, P.; LEPP, M. *RFC4098*: Terminology for Benchmarking BGP Device Convergence in the Control Plane. RFC Editor, 2005. Category: Informational. (Network Working Group. Request for Comments, 4098). Disponível em: <http://tools.ietf.org/html/rfc4098>. Acesso em: agosto/2013.

BHATIA, S.; MOTIWALA, M.; MUHLBAUER, W.; VALANCIUS, V.; BAVIER, A.; FEAMSTER, N.; PETERSON, L.; REXFORD, J. Hosting Virtual Networks on Commodity Hardware. In: *Techinal Report GT-CS-07-10, January 2008*. [S.l.]: Georgia Tech. University, 2008.

BLUNK, L.; KARIR, M.; LABOVITZ, C. *RFC6396*: Multi-Threaded Routing Toolkit (MRT) Routing Information Export Format. RFC Editor, 2011. Category: Standards Track. (Internet Engineering Task Force. Request for Comments, 6396). Proposed Standard. Errata Exist. Disponível em: <http://tools.ietf.org/html/rfc6396>. Acesso em: Dezembro/2013.

BONAVENTURE, O.; FILSFILS, C.; FRANCOIS, P. Archieving Sub-50 Milliseconds Recovery Upon BGP Peering Link Failures. In: . [S.l.]: IEEE. Institute of Electrical and Electronics Engineers, 2007. v. 15, n. 5, p. 1123–1135.

BRAUN, W.; MENTH, M. Wildcard Compression of Inter-Domain Routing Tables for OpenFlow-Based Software-Defined Networking. *Future Internet*, Multidisciplinary Digital Publishing Institute, v. 6, n. 2, p. 302–336, 2014.

CÁCERES, R.; DANZIG, P. B.; JAMIN, S.; MITZEL, D. J. Characteristics of Wide-area TCP/IP Conversations. *SIGCOMM Computer Communication Review*, ACM, Association for Computing Machinery, New York, NY, USA, v. 21, n. 4, p. 101–112, august 1991. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/115994.116003>.

CHEN, E.; REKHTER, Y. *RFC5291*: Outbound Route Filtering Capability for BGP-4. RFC Editor, 2008. Category: Standards Track. (Network Working Group. Request for Comments, 5291). Disponível em: <http://tools.ietf.org/html/rfc5291>. Acesso em: agosto/2013.

CISCO Systems. *Cisco Express Forwarding Overview*. [S.l.], 2012. (Release 12.2. Cisco Systems). Cisco IOS Switching Services Configuration Guide. Disponível em: <http://www.cisco.com/en/US/docs/ios/12.2/switch/configuration/guide/xcfcef.html>. Acesso em: 09/2013.

COMER, D. E. Interligação de redes com TCP/IP: Princípios, protocolos e arquiteturas. In: ____. 5. ed. Rio de Janeiro: Elsevier, 2006. v. 1, brochura. Capítulo 9 - Extensões de endereçamento classless e de sub-rede (CIDR), p. 81–97. ISBN 9788535220179. Tradução Daniel Vieira.

DRAVES, R. P.; KING, C.; VENKATACHARY, S.; ZILL, B. N. Constructing optimal IP routing tables (ORTC). In: *Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*. New York, NY: IEEE. Institute of Electrical and Electronics Engineers, 1999. (INFOCOM'99, v. 1), p. 88–97. ISBN 0-7803-5417-6. ISSN 0743-166X. Disponível em: <http://dx.doi.org/10.1109/INFCOM.1999.749256>.

FALL, K. R.; GODFREY, B.; IANNACCONE, G.; RATNASAMY, S. Routing Tables: Is Smaller Really Much Better? In: SUBRAMANIAN, L.; LELAND, W. E.; MAHAJAN, R. (Ed.). *Hot Topics in Networks VIII*. [s.n.], 2009. (HotNets '09), p. 22–23. Disponível em: <http://conferences.sigcomm.org/hotnets/2009/papers/hotnets2009-final156.pdf>.

FARIAS, L. F. T. de; DINIZ, M. C.; LUCENA, S. C. de; CORRÊA, C. N. A. Um Estudo Comparativo de Softwares de Roteamento para Uso em Redes Definidas por Software. In: *XXXIII Workshop em Desempenho de Sistemas Computacionais e de Comunicação*. [S.l.]: SBC - Sociedade Brasileira de Computação, 2013.

FULLER, V.; LI, T. *RFC4632*: Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan. RFC Editor, 2006. Category: Best Current Practice. (Network Working Group. Request for Comments, 4632). Disponível em: <http://tools.ietf.org/html/rfc4632>. Acesso em: junho/2013.

GUEDES, D.; VIEIRA, L. F. M.; VIEIRA, M. M.; RODRIGUES, H.; NUNES, R. V. Redes Definidas por Software: uma abordagem sistêmica para o desenvolvimento de pesquisas em Redes de Computadores. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2012*, SBC Sociedade Brasileira de Computação, v. 30, n. 4, p. 160–210, 2012.

GUPTA, A.; VANBEVER, L.; SHAHBAZ, M.; DONOVAN, S. P.; SCHLINKER, B.; FEAMSTER, N.; REXFORD, J.; SHENKER, S.; CLARK, R.; KATZ-BASSETT, E. SDX: A Software Defined Internet Exchange. In: *Proceedings of the 2014 ACM*

Conference on SIGCOMM. New York, NY, USA: Association for Computing Machinery, 2014. (SIGCOMM '14), p. 551–562. ISBN 978-1-4503-2836-4. Disponível em: <http://doi.acm.org/10.1145/2619239.2626300>.

HANDLEY, M.; HODSON, O.; KOHLER, E. XORP: An Open Platform for Network Research. *SIGCOMM Computer Communication Review*, Association for Computing Machinery, New York, NY, USA, v. 33, n. 1, p. 53–57, January 2003. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/774763.774771>.

HE, L. Building Next Generation, Open, Secure, Reliable and Scalable Experimental IP networks. In: . [S.l.]: Networking, IEEE/ACM Transactions on, 2005. v. 13, n. 6.

HOPPS, C. E. *RFC2992*: Analysis of an Equal-Cost Multi-Path Algorithmics. RFC Editor, 2000. Category: Informational. (Network Working Group. Request for Comments, 2992). Updated by RFC 5227, RFC 5494. Also Known As STD 37. Disponível em: <http://tools.ietf.org/html/rfc2992>. Acesso em: março/2014.

HUSTON, G. *BGP Routing Table Analysis Reports*. Australia, 2013. Online. Personal site of Geoff Huston provides updated statistical analysis and reports on Internet BGP data. This site is supported by APNIC. Disponível em: <http://bgp.potaroo.net>. Acesso em: 09/2013.

HUSTON, G.; SMITH, P.; BATES, T. *CIDR Report analyses the BGP Routing Table*. Online, 2014. Personal site of Geoff Huston provides updated statistical analysis and reports on Internet BGP data. This site is supported by APNIC. Disponível em: <http://www.cidr-report.org/as2.0/>. Acesso em: 24/06/2014.

IANA. *CCTLD News*. 2014. In Memo #1 - 23/10/1997. Iana - Internet Assigned Numbers Authority. Online. Acessado em Março/2014. Disponível em: `https://www.iana.org/reports/1997/cctld-news-oct1997.html`.

KALISZAN, A.; GŁĄBOWSKI, M.; HANCZEWSKI, S. A Didactic Platform for Testing and Developing Routing Protocols. In: *The Eighth Advanced International Conference on Telecommunications*. Stuttgart, Germany: IARIA, International Academy, Research, and Industry Association, 2012. (AICT 2012), p. 197–202. ISBN 978-1-61208-199-1. ISSN 2308-4030. Disponível em: <http://www.thinkmind.org/index.php?view=article&articleid=aict_2012_8_40_10218>.

KANAUMI, Y.; SAITO, S.; KAWAI, E.; ISHII, S.; KOBAYASHI, K.; SHIMOJO, S. Deployment and operation of wide-area hybrid OpenFlow networks. In: *13th IEEE/IFIP Network Operations and Management Symposium*. Maui, HI. USA: IEEE. Institute of Electrical and Electronics Engineers, 2012. (NOMS'12, v. 13), p. 1135–1142. ISBN 978-1-4673-0267-8. ISSN 1542-1201.

KLEINROCK, L.; KAMOUN, F. Hierarchical routing for large networks performance evaluation and optimization. *Computer Networks 1976*, Elsevier B.V., Atlanta, GA, USA, v. 1, n. 3, p. 155–174, January 1977. ISSN 0376-5075. This research was supported by the Advanced Research Projects Agency of The Department of Defense under Contract DAHC 15-73-C-0368. Disponível em: <http://www.sciencedirect.com/science/article/pii/0376507577900022>.

KUMAR, M.; KUMAR, S. Improving routing in large networks inside autonomous system. In: . [S.l.: s.n.], 2013. p. 1–8. Springer India.

LIU, Y.; AMIN, S. O.; WANG, L. Efficient FIB Caching using Minimal Non-overlapping Prefixes. *SIGCOMM Computer Communication Review*, ACN, Association for Computing Machinery, New York, NY, USA, v. 43, n. 1, p. 14–21, January 2013. ISSN 0146-4833. This work was supported by NSF Grant 0721645. Disponível em: <http://doi.acm.org/10.1145/2427036.2427039>.

LIU, Y.; ZHANG, B.; WANG, L. FIFA: Fast incremental FIB aggregation. In: IEEE. Institute of Electrical and Electronics Engineers. *2013 Proceedings IEEE*. Turim, Itália, 2013. (INFOCOM, 2013, v. 32), p. 1–9. ISBN 978-1-4673-5944-3. ISSN 0743-166X.

MAI, J.; DU, J. BGP performance analysis for large scale VPN. In: *Information Science and Technology (ICIST), 2013 International Conference on*. [S.l.: s.n.], 2013. p. 722–725.

MANRAL, V.; WHITE, R.; SHAIKH, A. *RFC4061*: Benchmarking Basic (OSPF) Single Router Control Plane Convergence. RFC Editor, 2005. Category: Informational. (Network Working Group. Request for Comments, 4061). Disponível em: <http://tools.ietf.org/html/rfc4061>. Acesso em: junho/2012.

MCKEOWN, N.; ANDERSON, T.; BALAKRISHNAN, H.; PARULKAR, G. M.; PETERSON, L. L.; REXFORD, J.; SHENKER, S.; TURNER, J. OpenFlow: Enabling Innovation in Campus Networks. In: . New York, NY, USA: ACM. Association for Computing Machinery, 2008. v. 38, n. 2, p. 69–74. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1355734.1355746>.

MEYER, D.; ZHANG, L.; FALL, K. *RFC4984*: Report from the IAB Workshop on Routing and Addressing. RFC Editor, 2007. Category: Informational. (Internet Engineering Task Force. Request for Comments, 4984). Errata Exist. Disponível em: <http://tools.ietf.org/html/rfc4984>. Acesso em: Dezembro/2013.

MEYER, D. M. *RouteViews project*. Advanced Network Technology Center, 2014. On line. Disponível em: <http://www.routeviews.org/.> Acesso em: Março/2014.

MOHAMED, H.; BASHANDY, A.; SHAHEEN, S. I. Protocol based selective FIB download for distributed forwarding architecture. In: IEEE. Institute of Electrical and Electronics Engineers. *Global Communications Conference Workshops (GCC Wkshps)*. Anaheim, CA, USA: Communications Society (ComSoc), 2012. p. 7–12. ISBN 978-1-4673-4942-0. Disponível em: <http://www.ieee-globecom.org/2012>.

MOORE, G. E. Cramming more components onto integrated circuits. In: *Electronic Design Magazine*. New York, NY: Penton Media, 1965. v. 38, n. 8. ISSN 0013-4872.

MOREIRA, M. D. D.; FERNANDES, N. C.; COSTA, L. H. M. K.; DUARTE, O. C. M. B. Internet do futuro: Um novo horizonte. In: *27o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. [S.l.]: SBC Sociedade Brasileira de Computação, 2009.

MORRISON, D. R. PATRICIA—Practical Algorithm To Retrieve Information Coded in Alphanumeric. *JACM, Journal of the ACM*, Association for Computing Machinery, New

York, NY, USA, v. 15, n. 4, p. 514–534, October 1968. ISSN 0004-5411. Disponível em: <http://doi.acm.org/10.1145/321479.321481>.

NARTEN, T.; NORDMARK, E.; SIMPSON, W. A.; SOLIMAN, H. *RFC4861*: Neighbor Discovery for IP version 6 (IPv6). RFC Editor, 2007. Category: Standards Track. (Network Working Group. Request for Comments, 4861). Updated by RFC 5942, RFC 6980, RFC 7048. Obsoletes RFC 2461. Was draft-ietf-ipv6-2461bis (ipv6 WG). Disponível em: <http://tools.ietf.org/html/rfc4861>. Acesso em: junho/2014.

NASCIMENTO, M. R.; ROTHENBERG, C. E.; SALVADOR, M. R.; MAGALHÃES, M. F. RouteFlow: Roteamento Commodity Sobre Redes Programáveis. In: *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. [S.l.]: SBC Sociedade Brasileira de Computação, 2011.

ONDREJ, F.; MARES, M.; ONDREJ, Z. The BIRD Internet Routing Daemon. In: CZ.NIC LABS. [S.l.], 2012. Acessado em Setembro (2012), `http://bird.network.cz`.

PLUMMER, D. C. *RFC826*: An Ethernet Address Resolution Protocol. RFC Editor, 1982. Category: Internet Standards. (Network Working Group. Request for Comments, 826). Updated by RFC 5227, RFC 5494. Also Known As STD 37. Disponível em: <http://tools.ietf.org/html/rfc826>. Acesso em: agosto/2013.

RASZUK, R.; HEITZ, J.; LO, A.; ZHANG, L.; XU, X. *RFC6769*: Simple Virtual Aggregation (S-VA). RFC Editor, 2012. Category: Informational. (Internet Engineering Task Force. Request for Comments, 6769). Disponível em: <http://tools.ietf.org/html/rfc6769>. Acesso em: junho/2013.

ROSEN, E. C.; REKHTER, Y. *RFC4364*: BGP/MPLS IP Virtual Private Networks (VPNs). RFC Editor, 2006. Category: Standards Track. (Network Working Group. Request for Comments, 4364). Updated by RFC 5462, RFC 4684, RFC 4577 Obsoletes RFC 2547. Was draft-ietf-l3vpn-rfc2547bis (l3vpn WG). Disponível em: <http://tools.ietf.org/html/rfc4364>. Acesso em: novembro/2013.

ROSEN, E. C.; VISWANATHAN, A.; CALLON, R. *RFC3031*: Multiprotocol Label Switching Architecture. RFC Editor, 2001. Category: Standards Track. (Network Working Group. Request for Comments, 3031). Updated by RFC 6790, RFC 6178. Was draft-ietf-mpls-arch (mpls WG). Disponível em: <http://tools.ietf.org/html/rfc3031>. Acesso em: outubro/2013.

ROTHENBERG, C.; NASCIMENTO, M.; SALVADOR, M.; CORRÊA, C.; LUCENA, S.; ALLAN, V.; VERDI, F. L. Revisiting IP Routing Control Platforms with OpenFlow-based Software-Defined Networks. In: *III Workshop de Pesquisa Experimental da Internet do Futuro*. [S.l.: s.n.], 2012.

ROTTENSTREICH, O.; RADAN, M.; CASSUTO, Y.; KESLASSY, I.; ARAD, C.; MIZRAHI, T.; REVAH, Y.; HASSIDIM, A. Compressing forwarding tables. In: *INFOCOM, 2013*. [S.l.]: IEEE, 2013. p. 1231–1239. ISSN 0743-166X.

SÁNCHEZ, B. C. *Virtual Aggregation in OpenFlow Networks*. 74 p. Dissertação (Student thesis) — KTH, School of Information and Communication Technology (ICT), Stockholm,

Sweden, Maio 2013. Educational program: Master of Science - Communication Systems. Disponível em: <http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-121765>.

SANGHANI, B. *2011 Report on European IXPs*. Amsterdam, The Netherlands, 2012. Euro-IX IXP Report Series. Disponível em: <https://www.euro-ix.net/documents/1024-Euro-IX-IXP-Report-pdf>.

SHAHZAD, S. A. *Route aggregation in Software-defined Networks*. 46 p. Dissertação (Student thesis) — KTH, School of Information and Communication Technology (ICT), Stockholm, Sweden, Junho 2013. Educational program: Master of Science - Communication Systems. Disponível em: <http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-123977>.

SINHA, R.; ZOBEL, J. Cache-conscious sorting of large sets of strings with dynamic stries. *JEA, Journal of Experimental Algorithmics*, Association for Computing Machinery, New York, NY, USA, v. 9, n. 1.5, p. 1–5, December 2004. ISSN 1084-6654. Disponível em: <http://doi.acm.org/10.1145/1005813.1041517>.

TANENBAUM, A. S. Organização Estruturada de Computadores. In: ____. 4. ed. Rio de Janeiro: Prentice Hall, 1999. v. 0, brochura capítulo 2, p. 38–40. ISBN 9788521612537. Tradução Nery Machado Filho.

TROSVIK, H. *Open Source Routing Suites*. iLabs Open Source Software Initiative, 2006. White Paper. (InteropNet). Disponível em: <http://www.interop.com/lasvegas/exhibition/interoplabs/2006/oss/opensourcerouting.pdf>. Acesso em: junho/2013.

TROTTER, G. *RFC3222*: Terminology for Forwarding Information Base (FIB) based Router Performance. RFC Editor, 2001. Category: Informational. (Network Working Group. Request for Comments, 3222). Disponível em: <http://tools.ietf.org/html/rfc3222>. Acesso em: agosto/2013.

UZMI, Z. A.; NEBEL, M.; TARIQ, A.; JAWAD, S.; CHEN, R.; SHAIKH, A.; WANG, J.; FRANCIS, P. SMALTA: practical and near-optimal FIB aggregation. In: *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*. New York, NY, USA: Association for Computing Machinery, 2011. (CoNEXT '11), p. 29:1–29:12. ISBN 978-1-4503-1041-3. Disponível em: <http://doi.acm.org/10.1145/2079296.2079325>.

ANEXO A – Rotinas de Extração

```
#!/bin/bash
# ##################################### LUFTF ###
# DESCRIÇÃO: Dados da RIB são armazenados em 2 Arquivos para uso
#                          no script de agregação.
# #######
declare -A IFACE MASK

# ####### Faz a verredura da FIB na SR ###
ip route show | grep via | awk '{print $1"\t"$3"\t"$5"\t"$7}' > tmp
sort -t . -k1,1n -k2,2n -k 3,3n -k 4,4n tmp > $ARQSAI.completo

# ####### Indexa vetor numérico NET[*] ###
NET=( $(awk '{print $1}' $ARQSAI.completo | cut -d "/" -f 1) )
iface=( $(awk '{print $3}' $ARQSAI.completo) )
mask=( $(awk '{print $1}' $ARQSAI.completo | cut -d "/" -f 2) )
lin=0
for i in $(echo ${NET[@]}); do
     IFACE+=(["$i"]="${iface[$lin]}")  # Preenche vetor associativo
     MASK+=(["$i"]="${mask[$lin]}")
     let lin=$lin+1
done

# ####### Faz a ordenação dos registros ###
declare -A IFACE
ARQ_ROTAS="$1" # Nome do arquivo com a tabela de rotas
F_PREFIXO=1         # Coluna no arq-ROTAS com os prefixos de rede
F_IFACE=3      # Coluna no arq-ROTAS com as interfaces
sort -t . -k 1,1n -k 2,2n -k 3,3n -k 4,4n $ARQ_ROTAS > temp
IPS=( $(awk -v n=${F_PREFIXO} '{print $n}' temp | cut -d "/" -f 1 ) )

# ####### Preenche vetor associativo ###
iface=( $(awk -v n=${F_IFACE} '{print $n}' temp ) ); lin=0
for i in $(echo ${IPS[@]}); do
     IFACE+=(["$i"]="${iface[$lin]}")
     let lin=$lin+1
done
unset iface
for key in $(echo ${IPS[@]}); do
     echo -ne "${IFACE[$key]}\t\t|${IFACE[$key]}"
done
```

ANEXO B – Rotinas de Redução

```
#!/bin/bash
# ##################################### LUFTF ###
declare -A IFACE AGG IBIT   # Vetores associativos

function PREFIXAR(){
     # ###################################
     # Cria o prefixo do endereço IP usando a posição bit
     # Parâmetros: Endereço IP e BIT usado para corte
     # #######
     local prefixo=${1} #... Endereço IP que será prefixado
     local ibit=${2}     #... BIT usado para corte
# ####### Separação e seleção dos octetos ###
     oct1=$(echo $prefixo | cut -d "." -f 1 )
     oct2=$(echo $prefixo | cut -d "." -f 2 )
     oct3=$(echo $prefixo | cut -d "." -f 3 )
     oct4=$(echo $prefixo | cut -d "." -f 4 )
     if [ $ibit -le 8 ]; then pref_fixo=""
          octalvo=$oct1
          suf_fixo=".0.0.0"
          let qtdbits_octalvo=$ibit;
     elif [ $ibit -le 16 ]; then pref_fixo="$oct1."
          octalvo=$oct2
          suf_fixo=".0.0"
          let qtdbits_octalvo=$ibit-8;
     elif [ $ibit -le 24 ]; then pref_fixo="$oct1.$oct2."
          octalvo=$oct3
          suf_fixo=".0"
          let qtdbits_octalvo=$ibit-16;
     else pref_fixo="$oct1.$oct2.$oct3."
          octalvo=$oct4
          suf_fixo=""
          let qtdbits_octalvo=$ibit-24;
     fi
# ####### Formação do prefixo/sufixo em binário ###
     octalvobin="$(printf '%08d\n' $(echo "obase=2;$octalvo" | bc) )"
     if [ $qtdbits_octalvo -eq 0 ];
```

```
          then prefbin_octpref="$octalvobin";
          else prefbin_octpref="${octalvobin::$qtdbits_octalvo}";
     fi
     let nzeros=8-$qtdbits_octalvo
     zeros="00000000"
     if [ $nzeros -eq 8 ];
          then sufbin_octpref="";
          else sufbin_octpref="${zeros::$nzeros}";
     fi
# ####### Montagem do IP após união do octeto em binário ###
     octprefbin=$prefbin_octpref$sufbin_octpref
     octpref=$((2#$octprefbin))
     IPPREF=$pref_fixo$octpref$suf_fixo
     echo $IPPREF
}
function VERIFICASTATUS(){
     for inet in $(echo ${NET[@]}); do
          istatus=$(echo ${STATUS[$inet]})
          if [ $istatus == "desagg" ]; then
               desaggNET=(${desaggNET[@]} "$inet");
          fi
     done
}
function PROXBIT(){
     local ipdec=${1}
     local ibit=${2}
     local M=($@)
     for i in 1 2 3 4; do
          octdec=$(echo ${ipdec} | cut -d "." -f $i )
          octbin=$(printf '%08d\n' $(echo "obase=2;$octdec" | bc) )
          ipbin=$(echo $ipbin$octbin)
     done
     restosufixo=$(echo ${ipbin:$ibit})
     POSBIT1=$(expr index "$restosufixo" 1)
     let IBIT=$ibit+$POSBIT1
     echo $IBIT
}

# ##################################### LUFTF ###
# ROTINA PRINCIPAL - Indexa vetores NET[*] com os prefixos da RIB
# #######
```

```
NET=( $(awk -v n=${F_NET} '{print $n}' $ARQ_FIB) )
iface=( $(awk -v n=${F_IFACE} '{print $n}' $ARQ_FIB) )
lin=0

for i in $(echo ${NET[@]}); do
     IFACE+=(["$i"]="${iface[$lin]}")
     let lin=$lin+1
done
# ####### Seleção ###
IBIT=8; lin=0
for i in $(echo ${NET[@]}); do
     prefixo=$(PREFIXAR $i $IBIT)
     AGG+=(["$i"]="$prefixo")
     IBIT+=(["$i"]="$IBIT")
     let lin=$lin+1
done

# ####### FASE-1: Agrupamento ###
declare -A STATUS contagg contseq
for inet in ${NET[@]}; do
     iagg=$(echo ${AGG[$inet]})
     let contagg[$iagg]++
     iseq=$(echo $iagg.${IFACE[$inet]})
     let contseq[$iseq]++        # Acumulando os iguais iseq[1.0.0.0]=6
done

for inet in ${NET[@]}; do
    let totalagg=$(echo ${contagg[$(echo ${AGG[$inet]})]})
    let meioagg=$totalagg/2
    let totalseq=$(echo ${contseq[$(echo ${AGG[$inet]}.${IFACE[$inet]})]})
    if [ $totalseq -le $meioagg ];
        then STATUS+=(["$inet"]="desagg");
        elif [ $totalagg -eq $totalseq ]; then STATUS+=(["$inet"]="OK");
        else STATUS+=(["$inet"]="aguarde");
    fi
done

# ####### FASE-2:FILTRAGEM ###
unset desaggNET STATUS contagg contseq
declare -A STATUS contagg contseq
```

```
for inet in $(echo ${NET[@]});  do
     istatus=$(echo ${STATUS[$inet]})
     if [ $istatus == "desagg" ];
          then desaggNET=(${desaggNET[@]} "$inet");
     fi
done

while [ $(echo ${#desaggNET[*]}) -gt 0 ]; do
     for i in $(echo ${desaggNET[@]}); do
          ibitant=$(echo ${IBIT[$i]})
          unset IBIT["$i"]
          ibit=$(PROXBIT $i $ibitant)
          IBIT+=(["$i"]="$ibit")
          prefant=$(echo ${AGG[$i]})
          unset AGG["$i"]
          agregado=$(PREFIXAR $i ${IBIT[$i]})
          AGG+=(["$i"]="$agregado")
          unset STATUS["$i"]
     done

     for inet in ${NET[@]}; do
          iagg=$(echo ${AGG[$inet]})
          let contagg[$iagg]++
          iseq=$(echo $iagg.${IFACE[$inet]})
          let contseq[$iseq]++
     done

     for inet in ${NET[@]}; do
          let totalagg=$(echo ${contagg[$(echo ${AGG[$inet]})]})
          let meioagg=$totalagg/2
          let totalseq=$(echo ${contseq[$(echo ${AGG[$inet]}.${IFACE[$inet]})]})
          if [ $totalseq -le $meioagg ];
               then STATUS+=(["$inet"]="desagg");
               elif [ $totalagg -eq $totalseq ];
                    then STATUS+=(["$inet"]="OK");
               else STATUS+=(["$inet"]="aguarde");
          fi
     done
     unset desaggNET

     for inet in $(echo ${NET[@]}); do
```

```
          istatus=$(echo ${STATUS[$inet]})
          if [ $istatus == "desagg" ];
               then desaggNET=(${desaggNET[@]} "$inet"); fi
     done
done

# ####### FASE-3: Agregação ###
unset AGG IBIT IFACE
declare -A AGG IBIT IFACE STATUS contagg contseq
IBIT=8; lin=0
for i in ${PREFAGG[@]}; do
     IFACE+=(["$i"]="${DEV[$i]}")
done

for i in ${PREFAGG[@]}; do
     prefixo=$(PREFIXAR $i $IBIT)
     AGG+=(["$i"]="$prefixo")
     IBIT+=(["$i"]="$IBIT")
     let lin=$lin+1
done

for ndx in ${PREFAGG[@]}; do
     iagg=$(echo ${AGG[$ndx]})
     let contagg[$iagg]++
     iseq=$(echo $iagg.${IFACE[$ndx]})
     let contseq[$iseq]++
done
```

ANEXO C – Rotinas de Manutenção

```
# ##################################### LUFTF ###
# configura BIRD
# #######
declare -A _MACS[IFACES] _IPS[IFACES] _MASKS[IFACES]
declare -A PREFS_FULLRIB[IFACE] GW_VIZINHO[IFACE]
SR="bird"
qtdlinhas=$(wc -l listarouter | awk '{print $1}')
linha=1   # ... primeira linha de leitura no arquivo listarouter

for iface in ${IFACES[*]}; do
     _MACS[$iface]=$(ip -o addr show dev $iface | awk '{print $13}')
     addrip4=$(ip -4 -o addr show dev $iface | head -1 | awk '{print $4}')
     _IPS[$iface]=$(echo $addrip4 | cut -d "/" -f 1)
     _MASKS[$iface]=$(echo $addrip4 | cut -d "/" -f 2)
done

for iface in ${IFACES[*]}; do
    PREFS_FULLRIB[$iface]=$(ip route show dev $iface | awk '{print $1}'\
    | sort -u)
    GW_VIZINHO[$iface]=$(ip route show dev $iface | grep via\
    | awk '{print $3}' | sort -u)
done

for ifaces in ${_IFACES[*]}; do
    ip link set $ifaces down
done
ip route flush cache

while [ $qtdlinhas -ge $linha ]
do
     ROUTERNAME=$(head -$linha listarouter | tail -1 | cut -d ":" -f 1)
     CONTEINER=$SR\_$ROUTERNAME
     tmplinha=$(grep $ROUTERNAME listarouter)
     QTDIFACE=$(echo $tmplinha | cut -d ":" -f 2)
# #######  Identifica o arquivo para configuração ###
     ARQCONF="/var/lib/lxc/$CONTEINER/rootfs/etc/bird.conf"
```

```
# ####### Configurando a ID e as funções device e kernel ###
     echo "router id $ROUTER;" > $ARQCONF    # Cria um novo arquivo
     echo " " >> $ARQCONF         # Acrescenta conteúdo
     echo "protocol device {"        >> $ARQCONF
     echo "        scan time 10;"        >> $ARQCONF
     echo "}" >> $ARQCONF
     echo " " >> $ARQCONF
     echo "protocol kernel {"          >> $ARQCONF
     echo "        scan time 10;"      >> $ARQCONF
     echo "        export all"         >> $ARQCONF
     echo "}" >> $ARQCONF
     echo " " >> $ARQCONF
router id $ROUTER;

protocol device {
        scan time 10;
}
# ####### Inicio do looping para configuração de cada interface ###
let qtdcoluna=$QTDIFACE*2+2
coluna=2    # referência p/1a coluna com a configuração de interface
while [ $coluna -lt $qtdcoluna ]
do
     let coluna=$coluna+1
     ETH=$(echo $tmplinha | cut -d ":" -f $coluna)
     IDETH='ETH'$ETH
     VETHPAIR=$ROUTERNAME\_$IDETH
     let coluna=$coluna+1
     IPSLASH=$(echo $tmplinha | cut -d ":" -f $coluna)
# ####### Configuração usada para cada interfaces ###
     echo " " >>  /var/lib/lxc/$CONTEINER/config
     echo "lxc.network.type = veth" >>  /var/lib/lxc/$CONTEINER/config
     echo "lxc.network.veth.pair = $VETHPAIR" >> /var/lib/lxc/$CONTEINER/config
     echo "lxc.network.link = br0" >>  /var/lib/lxc/$CONTEINER/config
     echo "lxc.network.ipv4 = $IPV4" >> /var/lib/lxc/$CONTEINER/config
     echo "lxc.network.name = $IDETH" >> /var/lib/lxc/$CONTEINER/config
     echo "lxc.network.flags = up" >>  /var/lib/lxc/$CONTEINER/config
     echo " " >>  /var/lib/lxc/$CONTEINER/config
done
let linha=$linha+1
done
```

ANEXO D – Criação Contêiner

```
# ##################################### LUFTF ###
# Criação do contêiner
# #######
SR="bird"
qtdlinhas=$(wc -l listarouter | awk '{print $1}')
linha=1
while [ $qtdlinhas -ge $linha ]
do
     ROUTERNAME=$(head -$linha listarouter | tail -1 | cut -d ":" -f 1)
     CONTEINER=$SR\_$ROUTERNAME
     tmplinha=$(grep $ROUTERNAME listarouter)
     QTDIFACE=$(echo $tmplinha | cut -d ":" -f 2)
     cp -pR /var/lib/lxc/"$SR" /var/lib/lxc/$CONTEINER 1>tmp
     rm tmp
# ####### Inicio do looping para configuração de cada interface ###
     let qtdcoluna=$QTDIFACE*2+2
     coluna=2
     echo "lxc.utsname = $CONTEINER" > /var/lib/lxc/$CONTEINER/config
     echo " " >> /var/lib/lxc/$CONTEINER/config
while [ $coluna -lt $qtdcoluna ]
do
     let coluna=$coluna+1
     ETH=$(echo $tmplinha | cut -d ":" -f $coluna)
     IDETH='eth'$ETH
     VETHPAIR=$ROUTERNAME\_$IDETH
     let coluna=$coluna+1
     IPSLASH=$(echo $tmplinha | cut -d ":" -f $coluna)
# ####### Configuração usada para cada interfaces ###
     echo "lxc.network.type = veth" >>  /var/lib/lxc/$CONTEINER/config
     echo "lxc.network.veth.pair = $VETHPAIR" >> /var/lib/lxc/$CONTEINER/config
     echo "lxc.network.link = br0" >>  /var/lib/lxc/$CONTEINER/config
     echo "lxc.network.ipv4 = $IPSLASH" >> /var/lib/lxc/$CONTEINER/config
     echo "lxc.network.name = $IDETH" >> /var/lib/lxc/$CONTEINER/config
     echo "lxc.network.flags = up" >>  /var/lib/lxc/$CONTEINER/config
     echo " " >>  /var/lib/lxc/$CONTEINER/config
done
```

```
# ####### Cria o arquivo de configuracao LXC para cada roteador ###
cat << EOF >> /var/lib/lxc/$CONTEINER/config
lxc.tty = 4
lxc.pts = 1024
lxc.rootfs = /var/lib/lxc/$CONTEINER/rootfs
lxc.cgroup.devices.deny = a
# /dev/null and zero
lxc.cgroup.devices.allow = c 1:3 rwm
lxc.cgroup.devices.allow = c 1:5 rwm
# consoles
lxc.cgroup.devices.allow = c 5:1 rwm
lxc.cgroup.devices.allow = c 5:0 rwm
lxc.cgroup.devices.allow = c 4:0 rwm
lxc.cgroup.devices.allow = c 4:1 rwm
# /dev/{,u}random
lxc.cgroup.devices.allow = c 1:9 rwm
lxc.cgroup.devices.allow = c 1:8 rwm
lxc.cgroup.devices.allow = c 136:* rwm
lxc.cgroup.devices.allow = c 5:2 rwm
# rtc
lxc.cgroup.devices.allow = c 254:0 rwm
# mounts point
lxc.mount.entry=proc /var/lib/lxc/$CONTEINER/rootfs/proc proc nodev,\
noexec,nosuid 0 0
lxc.mount.entry=devpts /var/lib/lxc/$CONTEINER/rootfs/dev/pts devpts\
defaults 0 0
lxc.mount.entry=sysfs /var/lib/lxc/$CONTEINER/rootfs/sys sysfs defaults 0 0
EOF

let linha=$linha+1
done
```