Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Faculdade de Engenharia

André Luís Dornelles Kern

Descoberta em topologias de redes privadas

Rio de Janeiro
2019

André Luís Dornelles Kern

Descoberta em topologias de redes privadas

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre em Engenharia Eletrônica, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Redes de Telecomunicações.

Orientador: Prof. D.Sc. Alexandre Sztajnberg
Orientador: Prof. D.Sc. Rodrigo de Souza Couto

Rio de Janeiro
2019

CATALOGAÇAO NA FONTE

UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

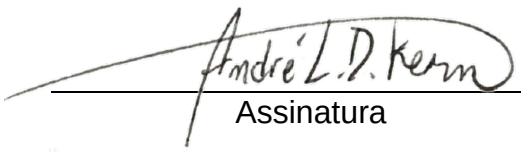
K39 Kern, André Luís Dornelles.
Descoberta em topologias de redes privadas / André Luís Dornelles Kern. – 2019.
72f.

Orientadores: Alexandre Sztajnberg, Rodrigo de Souza Couto.
Dissertação (Mestrado) – Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.

1. Engenharia eletrônica - Teses. 2. Redes de computadores - Teses. 3. OSI (Padrão de redes de computadores) - Teses. 4. Redes de computadores - Gerência - Teses. I. Sztajnberg, Alexandre. II. Couto, Rodrigo de Souza. III. Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia. IV. Título.

CDU 004.72

Bibliotecária: Júlia Vieira – CRB7/6022



Assinatura

27 de Fevereiro de 2019

Data

André Luís Dornelles Kern

Descoberta em topologias e redes privadas

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre em Engenharia Eletrônica, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Redes de Telecomunicações.

Aprovada em 27 de Fevereiro de 2019.
Banca Examinadora:

Prof. D.Sc. Alexandre Sztajnberg (Orientador)
PEL/UERJ

Prof. D.Sc. Rodrigo de Souza Couto (Orientador)
PEE/COPPE/UFRJ

Prof. D.Sc. Fabiano de Souza Oliveira
DICC/IME/UERJ

Prof. D.Sc. Miguel Elias M. Campista
PEE/COPPE/UFRJ

Rio de Janeiro
2019

DEDICATÓRIA

Aos meus pais, irmãos e dois filhos adorados por todo carinho que recebi durante a elaboração deste trabalho, me ajudando a chegar até esta etapa de minha vida. Dedico também a todos aqueles que de alguma forma estiveram e estão próximos de mim, fazendo esta vida valer cada vez mais a pena.

AGRADECIMENTOS

Agradeço aos Professores. Rodrigo Couto e Alexandre Sztajnberg pelas orientações, grande desprendimento em ajudar e amizade sincera, que muitas vezes sacrificaram suas horas de descanso para esclarecerem minhas dúvidas, ofereceram recursos para conseguir vencer os desafios e tornaram possível a conclusão desta dissertação. Agradeço ao meu amigo Rômulo Gon Ferreira por sua parceria nos estudos e sua influencia positiva durante o mestrado. Agradeço aos amigos de trabalho Carlos Teles e Diego Aguirre que me ajudaram, por meio de sugestões, a conseguir resultados mais rapidamente. Agradeço aos meus pais e irmão que muitas vezes cederam seus computadores para que eu conciliasse a família e a pesquisa do mestrado. Agradeço aos meus filhos que, apesar da pouca idade, se mostraram pacientes e sempre tinham uma palavra carinhosa para me animar.

Heróis trabalham para realizar seus projetos de vida, não para impressionar os outros.

Roberto Shinyashiki

RESUMO

KERN, A. L. D. *Descoberta* em *topologia*s de *redes privadas*. 2019. 72 f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2019.

O aumento da produção e demanda de conteúdos digitais, que utilizam redes de computadores como meio de transferência, leva ao crescimento da infraestrutura dessas redes. O monitoramento dos dispositivos de rede ativos, por sua vez, é necessário para prover serviço resiliente de qualidade. Assim, a identificação rápida e definida de possíveis falhas ou alterações na rede depende do mapeamento dos nós participantes nas decisões de encaminhamento bem como dos enlaces de rede. A descoberta de sub-rede é um conceito de camada de enlace, representada pela camada 2 (*layer 2*, L2) do modelo OSI (*Open Systems Interconnection*), que tem como propósito mapear a topologia por meio do agrupamento das interfaces de redes interconectadas em um mesmo domínio de colisão. Nas técnicas analisadas de descoberta de topologia, a localização de dispositivos L2 apresenta maior complexidade pela dificuldade de identificação deles entre os dispositivos de rede, sendo os dispositivos L2 mais numerosos que dispositivos L3 em redes privadas de grande porte. Este trabalho tem como objetivo explorar essa dificuldade através do mapeamento de rede com a técnica que utiliza o algoritmo *Skeleton-tree*, alterando método de coleta de dados da técnica para diminuir o tempo total utilizado na descoberta de topologia. A alteração do método ativo de ICMP (*Internet Control Message Protocol*) para ARP (*Address Resolution Protocol*) apresenta maior velocidade no processo completo de descoberta de topologia de rede, decorrente da melhor eficiência na atualização e coleta de dados das AFTs (*Address Forwarding Tables*) em cenários com tempo limite de espera (timeout) reduzido. Essa alteração resulta em base de dados de entrada de melhor qualidade para o algoritmo *Skeleton-tree*, resultando em menor probabilidade de falha na implementação da descoberta de topologia.

Palavras-chave: Descoberta de topologia; Camada 2; Infraestrutura de rede.

ABSTRACT

KERN, A. L. D. P*rivate network topology discovery.* 2019. 72 f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2019.

The increase in the production and demand of digital content, which uses computer networks as a mean of transfer, leads to the growth of the infrastructure of these networks. The monitoring of active network devices, in turn, is necessary to provide resilient quality service. Thus, the rapid and defined identification of possible network failures or changes depends on the mapping of the nodes involved in the forwarding decisions as well as the network links. Subnet Discovery is a concept of the link layer, represented by layer 2 (L2) of the OSI (*Open Systems Interconnection*) model, whose purpose is to map the topology through the interconnected network interfaces in the same broadcast domain. In the analyzed techniques of topology discovery, the location of L2 devices is more complex due to the difficulty of identifying them among the network devices, with L2 devices being more numerous than L3 devices in large private networks. This work aims to explore this difficulty through network discovery with the technique that uses the *Skeleton-tree* algorithm, changing the method of data collection of the technique to decrease the total time used in topology discovery. Changing the active method from ICMP (*Internet Control Message Protocol*) to ARP (*Address Resolution Protocol*) presents greater speed in the complete network topology discovery process, resulting from better efficiency in updating and collecting AFT (*Address Forwarding Table*) data in reduced timeout scenarios. This change results in better-quality input database for the *Skeleton-tree* algorithm, resulting in a lower probability of failure in the implementation of topology discovery.

Keywords: Topology discovery; layer 2; Network infrastructure.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

AFT	*Address Forwarding Table*
API	*Application Programming Interface*
ARP	*Address Resolution Protocol*
AS	*Autonomous System*
BGP	*Border Gateway Protocol*
C2P	*Customer to Provider*
CDP	*Cisco Discovery Protocol*
GNS3	*Graphical Network Simulator-3*
ICMP	*Internet Control Message Protocol*
IDS	*Intrusion Detection System*
IEEE	*Institute of Electrical and Electronics Engineers*
IP	*Internet Protocol*
JSON	*JavaScript Object Notation*
L2	*Layer 2 - referente à camada 2 do modelo OSI*
LLDP	*Link Layer Discovery Protocol*
MAC	*Media Access Control*
MIB	*Management Information Base*
MSAP	*Media Service Access Point*
MST	*Multiple Spanning Tree Protocol*
NMS	*Network Management System*
OSI	*Open Systems Interconnection*
P2P	*Peer to Peer*
POC	*Proof of Concept*
PoP	*Point of Presence*
PVST	*Per VLAN Spanning Tree*
RAM	*Random Access Memory*
RFC	*Request for Comments*
RPVST	*Rapid Per VLAN Spanning Tree*
RSTP	*Rapid Spanning Tree Protocol*
STP	*Spanning Tree Protocol*
SMI	*Structure of Management Information*
TTL	*Time-To-Live*
VLAN	*Virtual LAN*
VM	*Virtual Machine*

SUMÁRIO

INTRODUÇÃO

O conhecimento da topologia de uma rede é fundamental para a rápida identificação dos recursos e caminhos de transferência de dados disponíveis. Manter informações atualizadas e ter uma visão acurada da topologia de uma rede é de interesse permanente em atividades de desenvolvimento e operação, seja para projetar, realizar avaliação de protocolos e serviços ou para a análise de vulnerabilidades em infraestruturas de rede. Em redes de grande porte, em que o crescimento e a evolução da infraestrutura podem ser administrados por equipes diferentes, mudanças podem ser realizadas de forma distribuída, o que torna desafiador manter uma visão consistente da topologia.

A segurança e a integridade de rede também dependem do conhecimento dos dispositivos a ela interconectados. Redes em que a segurança é uma característica primordial, a identificação rápida e eficiente de dispositivos não autorizados na topologia de rede pode ser um diferencial na realização de medidas de contenção. Um usuário da rede com o conhecimento adequado pode compartilhar sua interface, conectando um *hub* ou *switch* no lugar do seu computador, sem o consentimento dos administradores da rede. Nesse caso a execução do processo de descoberta de rede pode ser realizada de forma constante para fim de controle de segurança.

Outro ponto importante é o relacionado às alterações feitas na infraestrutura, que podem não serem devidamente documentadas pelos operadores. Isso resulta da falha na comunicação adequada das alterações aos indivíduos e sistemas que controlam os registros a topologia. Por exemplo, é necessário reconhecer dispositivos ativos conectados à infraestrutura de rede que não estão mais em uso depois de uma atualização de um dispositivo de rede por meio de substituição. O novo dispositivo é incluído na topologia, enquanto o antigo é mantido conectado por medida preventiva, em caso de necessidade de retorno para a estrutura original. Outro exemplo, que pode ser mencionado, é a instalação emergencial de dispositivo para expandir a disponibilidade de interfaces, que acabam tornando-se permanentes.

O mapeamento apurado de uma rede de grande porte também é necessário para os engenheiros e operadores que buscam constantemente aperfeiçoar a alocação de recursos. Técnicas de mapeamento de topologia de rede podem ser utilizadas em diversos níveis de detalhamento. As topologias resultantes são classificadas em diferentes níveis de detalhamento, podendo ser do nível-interface ao nível-AS (MOTAMEDI; REJAIE; WILLINGER, 2015).

Observa-se que as alterações em topologias de redes de grande porte se apresentam com maior frequência em detalhamento nível-interface, seja por necessidade de inclusão de recurso ou defeito em um dispositivo. Isso é compreensível dado que o nível-interface possui maior quantidade de dispositivos representados de forma individual. Por exemplo,

uma falha no nível-interface representa uma interface com problema e não o roteador completo. Essa falha detectada no nível-interface não é detectada em um nível de detalhamento mais abrangente. Uma visão menos detalhada e mais abrangente da rede é obtida à medida que se aproxima do nível-AS. Este assunto é aprofundado na Seção 1.3.

Para facilitar o mapeamento da topologia de uma rede, diversas técnicas de descoberta de topologia foram desenvolvidas com objetivo de realizar a identificação de dispositivos e de conexões de rede de forma sistematizada. Técnicas de descoberta de topologia deveriam ser capazes de identificar diversos tipos de dispositivos, como *hubs*, *switches* (comutadores), roteadores e *firewalls* e suas ligações. Estas técnicas em um primeiro momento coletam dados da rede e em seguida aplicam um algoritmo para montar a representação da topologia. A representação da topologia pode ser utilizada diretamente pela equipe de operação para a avaliação da rede e tomada de decisões, ou pode ser oferecida como entrada a ferramentas de gerência.

Técnicas simples e comumente utilizadas para descoberta de topologias na Internet empregam ferramentas como o *traceroute* e informações de tabelas de rota como as construídas pelo BGP (*Border Gateway Protocol*) para alcançar sistemas autônomos. O *traceroute* [1] é uma ferramenta de depuração usada para identificar quais nós das redes são atravessados para atingir um determinado destino. O BGP é o protocolo de o roteamento entre domínios (*inter-domain*) utilizado na Internet global e indica a alcançabilidade dos Sistemas Autônomos (*Autonomous System*, AS). A identificação dos roteadores por onde um pacote passa ou das áreas transpostas no BGP representam o caminho completo que uma rota precisa percorrer para chegar da origem até o destino (MOTAMEDI; REJAIE; WILLINGER, 2015).

Observa-se que, com essas técnicas utilizadas de forma direta, é possível obter apenas a topologia de um ponto de visão específico da rede. As informações utilizadas em um determinado nível de detalhamento podem não ser úteis em outro nível. A identificação de um AS BGP, por exemplo, não auxilia na descoberta de dispositivos no nível-interface. Para ampliar a abrangência , detectando a maior quantidade de nós possíveis, é necessário sistematizar de alguma forma a coleta de informações para todas as partes da rede.

Técnicas de descoberta mais completas automatizam a descoberta de topologias e podem utilizar métodos específicos, diferentes dos mencionados anteriormente, para explorar a rede e coletar os dados. Por exemplo, a descoberta de topologias com o uso do CDP (*Cisco Discovery Protocol*) é uma técnica que se mostra capaz de descobrir a topologia física da rede, fazendo uso dos dados do CDP em conjunto com as informações do *spanning-tree*. Entretanto, a técnica não se apresenta eficiente em redes heterogêneas com fabricantes não Cisco por fazer uso de protocolo proprietário (RODRIGUEZ, 2009).

[1] https://tools.ietf.org/html/rfc1393

Tabela 1. Camadas Modelo OSI de referência.

7 Aplicação	interface com aplicações que necessitam acesso a rede
6 Apresentação	interpreta dados para a camada de sessão ou aplicação
5 Sessão	estabelece serviços de segurança e identificação do usuário
4 Transporte	segmenta, controla fluxo e transporta dados
3 Rede	fornece endereçamento lógico único e roteamento
2 Enlace	prepara pacotes da camada de rede para o meio físico
3 Física	transmite sequências de bits pelo meio físico

Em outra técnica, o uso do *Link Layer Discovery Protocol* (LLDP) é apresentada como ferramenta para método de descoberta automática de topologia física em redes de automação. O LLDP é uma alternativa não proprietária ao CDP, com funcionamento definido no padrão 802.1ab, sendo uma alternativa em ambientes contendo dispositivos de diferentes fabricantes. Sua eficiência é comprometida quando se depara com dispositivos sem suporte ao LLDP (SCHAFER; FELSER, 2007). Extensões para o LLDP podem ser definidas para serem trocados informações não apenas entre vizinhos diretos, mas também em uma camada de enlace. Uma extensão proposta do protocolo permite o rastreamento de um endereço de hardware através de uma rede baseada na camada 2 do modelo OSI (SCHOLZ; KRUSE, 2015).

Modelo de referência da ISO, tem como principal objetivo ser um modelo padrão para protocolos de comunicação entre diversos tipos de sistema, garantindo a comunicação end-to-end, o Modelo OSI (em inglês Open Systems Interconnection) foi lançado em 1984 pela Organização Internacional para a Normalização (em inglês International Organization for Standardization).

O modelo OSI é uma arquitetura que divide as redes de computadores em 7 camadas para obter camadas de abstração. Tem como principal objetivo ser um modelo padrão para protocolos de comunicação entre diversos tipos de sistemas, garantindo a comunicação end-to-end. Cada protocolo realiza a inserção de uma funcionalidade assinalada a uma camada específica. Utilizando o Modelo OSI é possível realizar comunicação entre dispositivos distintas e definir instruções genéricas para a elaboração de redes de computadores independente da tecnologia utilizada, sejam essas redes de curta, média ou longa distância. Cada uma das 7 camadas realizam funções determinadas. As camadas são, em ordem decrescente: aplicação, apresentação, sessão, transporte, rede enlace e física, conforme descrito na Tabela 1.

A descoberta de topologia também pode se utilizar de técnicas que dependam da instalação de agentes nos nós finais. Tais técnicas geralmente não necessitam de acesso administrativo à rede para coleta de dados. Primeiramente, é realizada a descoberta dos dispositivos diretamente conectados aos nós finais. Com a referência dos dispositivos de primeiro nível, os outros dispositivos da rede são, então, descobertos. Esta técnica pode

ser utilizada para diagnóstico de rede, gerenciamento de recursos e implementação de servidor, porém não se aplica às topologias de complexidade maior (NOWICKI; MALINOWSKI, 2015). Essa complexidade maior representa uma rede com grande quantidade de elementos, níveis hierárquicos e conexões, podendo inviabilizar a instalação do total quantitativo de agentes necessários para a descoberta com a técnica.

Motivação

A abstração em nível-interface da topologia de rede retrata os dispositivos, suas interfaces IP e a conectividade entre elas. As interfaces IP de roteadores e *hosts* finais são representados como nós distintos. Esse nível de representação da topologia é importante para as equipes de gerência por auxiliar a identificar um dispositivo que necessite de atenção.

O primeiro desafio da descoberta da topologia em nível-interface é identificar a existência de dispositivos de rede que operam abaixo da camada 3 do modelo OSI. A dificuldade de identificar dispositivos L2 e L1 é comum entre as técnicas de descoberta de topologia, porque geralmente esses dispositivos não têm endereço lógico (IP) associado. Desta forma, nem sempre é possível utilizar ferramentas que dependam do IP, por exemplo, para a coleta de dados. Ignorar a existência de *switches* (L2) e *hubs* (L1) na rede, mesmo sendo considerados legados, pode levar a uma descoberta de topologia não acurada.

Um segundo desafio no mapeamento nível-interface é a descoberta de topologias com sub-redes, sendo esse um conceito de camada de enlace (L2) por meio do agrupamento das interfaces de sub-redes interconectadas em um mesmo domínio de difusão (*broadcast domain*). O estudo do domínio de difusão e dos dispositivos L2 pode aprimorar a clareza da topologia nível-interface, permitindo a visão dos dispositivos da rede local (LAN). Esta técnica pode se valer da identificação de caminhos L2, coletando dados definidos pelas AFTs e pelas informações de *spanning-tree*.

Objetivos

O objetivo inicial do nosso trabalho é obter uma técnica para descoberta de topologias que (i) atue em nível-interface e (ii) que consegue identificar corretamente dispositivos L2 e L1, e suas interconexões. A técnica obtida é avaliada quanto à probabilidade de falha, acurácia e ao tempo necessário para construir a representação de uma topologia.

Neste contexto, foram pesquisadas técnicas propostas para a descoberta de topologia que identificassem dispositivos abaixo da camada 3 (*layer 3* - L3) do modelo OSI,

incluindo dispositivos como *switches* não gerenciáveis (L2) e *hubs* (L1). Na pesquisa realizada, a influência do tempo na descoberta de uma topologia não foi levada em consideração, propondo-se neste trabalho uma abordagem diferente da literatura analisada.

Após pesquisa das técnicas e da avaliação de trabalhos relacionados (Capítulo 2), ainda foram incluídas duas características na técnica para a solução desejada: (iii) não deve depender de protocolo proprietário e (iv) não deve realizar modificação ou inclusão de dispositivos de hardware ou alteração de software.

Dentre os trabalhos relacionados, a técnica chamada *Skeleton-tree*, desenvolvida por Yigal Bejerano (BEJERANO, 2006) e (BEJERANO, 2009), chamou nossa atenção. A proposta inclui um algoritmo, denominado *Skeleton-tree*, que apresenta as características desejadas em uma técnica para descoberta de topologia, inferindo a existência de dispositivos em L1, L2 e L3. Esta proposta utiliza a ferramenta *ping* (ICMP-echo) como método ativo, combinado ao uso do SNMP como método passivo para coleta de dados e posterior execução do algoritmo *Skeleton-tree*. O protocolo SNMP coleta os dados utilizando o endereço IP do dispositivo alvo e o nome da *community* configurada. A *community* é o nome atribuído pelos administradores da rede que serve como identificador para requisição de informações dos dispositivos gerenciáveis.

Observa-se que Bejerano não disponibiliza informações sobre implementação, código ou a linguagem utilizada, apresentando somente uma discussão sobre o funcionamento do algoritmo. Além disso, o tempo de descoberta de uma topologia não é tratado.

O algoritmo *Skeleton-tree* foi, então, analisado com mais profundidade (Seção 3.1) para ser usado como base no desenvolvimento do nosso estudo. A partir da análise da técnica e do algoritmo de Bejerano, uma versão própria de código foi desenvolvida em *Python*, na qual foi implementado o algoritmo para descoberta de topologia em redes locais.

Observa-se que a técnica *Skeleton-tree* faz uso das informações da tabela de encaminhamento (*Address Forwarding Table* - AFT) dos dispositivos para descobrir a topologia. A AFT de cada dispositivo armazena as informações de encaminhamento, designando a interface que o dado deve ser encaminhado para alcançar o destino. Como parte de nossa investigação, propusemos o uso do ARP como alternativa ao ICMP, usado no processo de atualização das tabelas AFT da técnica original. Essa alteração teve como objetivo aprimorar o processo de preenchimento das tabelas AFT para depois se realizar a coleta de dados e a posterior descoberta de topologia de redes com o algoritmo. Assim, a descoberta de topologia também é avaliada em função do tempo total de execução. O aprimoramento se apresenta pelo menor tamanho e complexidade do pacote do ARP em relação ao ICMP e sua característica de alcançar todos os dispositivos por difusão. Perde-se com isso a comunicação individual com os dispositivos alvo, sendo essa associação feita pela estação que executa a varredura.

O protótipo desenvolvido é aplicado em ambiente de rede virtualizado no simulador

GNS3 para avaliação na sua opção original, usando ICMP, e com o uso do ARP, na topologia de referência e diversas topologias geradas de forma aleatória.

A implementação do protótipo do algoritmo original consegue de forma correta obter a topologia de redes com 120 dispositivos incluindo 10 dispositivos não gerenciáveis. Verifica-se, também, que utilizando a alternativa com o ARP o tempo total de descoberta é mais rápido e a descoberta mais precisa na maioria das topologias avaliadas, comparado com o uso do ICMP para atualização dos dados na proposta original.

Organização do Texto

A dissertação está estruturada da seguinte forma. O Capítulo 1 apresenta os conceitos básicos de descoberta de topologias de redes em diversos níveis de detalhamento, como a sua definição, dispositivos envolvidos, tecnologias relacionas e o GNS3 usado para as simulações. Os trabalhos relacionados e a posição desta dissertação em relação à literatura são tratados no Capítulo 2, onde são apresentadas diversas técnicas de descoberta de topologia com características distintas. No Capítulo 3 é apresentado o algoritmo *Skeleton-tree* e as modificações incluídas, sendo esmiuçados os passos do processo de descoberta, incluindo a atualização das tabelas AFTs, coleta dos dados, os detalhes do algoritmo e a implementação em *Python*. A forma como é feita a avaliação é apresentada no Capítulo 4, sendo abordado os recursos utilizados no ambiente de testes e as características das topologias analisadas. O Capítulo 5 apresenta os experimentos realizados, discute sobre os tempos de execução obtidos através da técnica, da heurística e avalia os resultados obtidos com a utilização do ARP e do ICMP em diversos cenários. Por fim, a dissertação é concluída no último capítulo, no qual são consideradas também as direções futuras.

1 CONCEITOS BÁSICOS

Este capítulo apresenta os principais conceitos sobre os dispositivos e recursos de topologias de rede. Então é destacado o uso dos recursos físicos e lógicos utilizados no desenvolvimento deste trabalho, incluindo os dispositivos e tecnologias relacionadas, a representação da topologia em seus níveis de detalhamento e o funcionamento do simulador GNS3. Atenção especial é concedido ao detalhamento nível-interface por ser o ponto de referência explorado no trabalho.

1.1 Dispositivos de uma Topologia de Rede

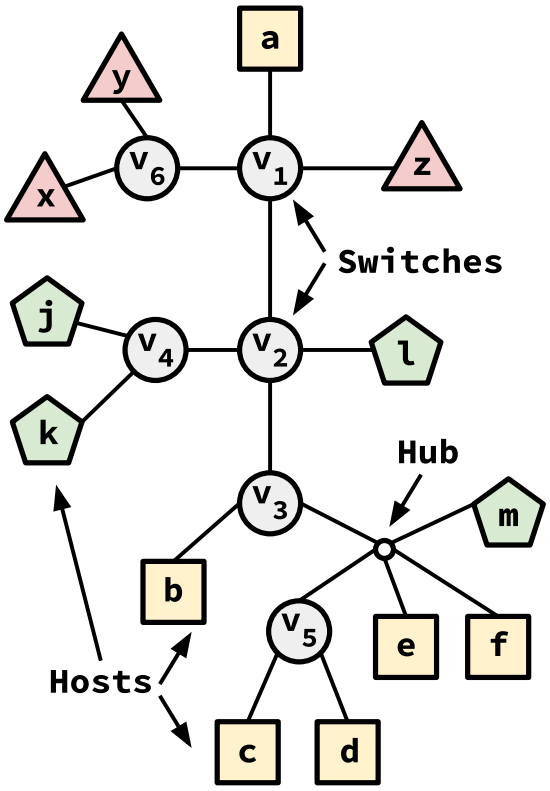
Topologia de rede refere-se ao layout físico ou lógico de uma rede. Ele define a maneira como diferentes nós são arranjados e interconectados entre si. Alternativamente, a topologia de rede pode descrever como os dados são transferidos entre esses nós. Existem dois tipos de topologias de rede: físicas e lógicas. A topologia física enfatiza o layout físico dos dispositivos e nós conectados, enquanto a topologia lógica se concentra no padrão de transferência de dados entre os nós da rede, indicando as conexões onde é possível a comunicação. Neste trabalho é abordado a topologia lógica, referente ao mapeamento dos nós e enlaces em função da transferência de dados, utilizando detalhamento nível-interface conforme descrito na Seção 1.3.

A delimitação de uma rede lógica leva em consideração o domínio no qual seus dispositivos estão inseridos. Dispositivos de camada de enlace, a camada 2 do modelo OSI (L2), se comunicam com demais dispositivos L2 desde que estejam em um mesmo domínio de difusão (*broadcast domain*) ARP. Desta maneira, os dispositivos formam uma rede caracterizada pelo seu endereço de acesso ao meio físico (*Media Access Control* - MAC).

Entretanto, a conectividade com sucesso em L2 não garante o mesmo sucesso quando a comunicação ocorre em L3. Por exemplo: um dispositivo pode iniciar uma comunicação ARP com sucesso para um destino, descobrindo seu endereço MAC (L2) e indicando sua existência. Em contrapartida pode não ter sucesso ao executar um *ping* para seu endereço IP (L3), o que leva à falsa impressão de inexistência do destino. Dessa forma, uma topologia formada por dispositivos L3 pode estar compreendida em uma topologia de dispositivos L2, porém a reciproca não ocorre obrigatoriamente.

A Figura 1 ajuda a ilustrar essa asserção. Na figura estão representadas as sub-redes definidas pelos elementos a ao f, j ao m, x ao z. Essas três sub-redes lógicas (L3) independentes coexistem em um mesmo domínio de difusão (L2), com seus dispositivos interconectados pelos *switches* e *hub*, conforme descrito na Seção 4.2. Na Seção 3.4 são

Figura 1. Redes lógicas da Topologia de Referência.



Fonte: Adaptado de (BEJERANO, 2009).

apresentadas as diferenças que podem ocorrer na descoberta da topologia com relação à detecção de dispositivos L2 e L3. Realizando a união de todos os caminhos entre dispositivos dois-a-dois resulta na topologia que representa as interconexões da rede. A topologia resultante é composta pela ordenação dos diversos dispositivos físicos e lógicos.

Dentre os principais dispositivos físicos intermediários podem ser mencionados roteadores, *switches*, *hubs* e *firewalls*. Esses dispositivos, a exceção do *hub*, podem ser implementados por meio de hardware dedicado ou por software em um computador comum. Neste trabalho são considerados como dispositivos intermediários os *switches* e *hubs* que realizam as conexões e definições de caminhos entre dois dispositivos finais. Os dispositivos de origem e destino da transferência de dados são considerados como dispositivos finais, aqui representados pelos *hosts*. A estação de gerenciamento de rede (*Network Management Station* - NMS), onde é gerenciado o processo de descoberta, também é considerada um dispositivo final.

Dentre os recursos lógicos podem ser mencionados os protocolos de definição de melhor caminho, com destaque neste trabalho ao protocolo *spannig-tree* (*Spanning Tree Protocol* - STP). O *ping* pode ser destacado como ferramenta que utiliza o protocolo ICMP para rastrear as interfaces de rede. O simulador de rede GNS3 é outro recurso lógico importante para o desenvolvimento deste trabalho, abordado com mais detalhes na

Seção 1.5.

Dispositivos comutadores, ou *switches*, são considerados dispositivos de camada de enlace e atuam na camada 2 (L2) do modelo OSI. Os *switches* são responsáveis por detectar erros e controlar troca de quadros, estabelecendo protocolo de comunicação entre sistemas diretamente conectados. *Switches* trabalham com tabelas de encaminhamento, as AFTs, populadas dinamicamente com endereços MAC e a porta de saída associada. As AFTs são formadas no domínio de difusão no qual o *switch* está inserido.

Dispositivos e recursos que trabalham na camada de rede (L3) e camadas superiores, como *firewalls*, IDS, IPS, e servidores em geral, são vistos como nós nas topologias com detalhamento nível-interface. Esses nós são identificados pelo seu endereço IP, não interferindo na descoberta de topologia, sendo tratados e identificados como *hosts* ou dispositivos finais.

1.2 Tecnologias Relacionadas

Neste trabalho, protocolos e utilitários são utilizados no suporte para a obtenção de dados, tratados na técnica do *Skeleton-tree* para a descoberta da topologia. Por exemplo, a existência do protocolo spanning-tree (*Spanning Tree Protocol* - STP) é necessária para que a técnica consiga identificar enlaces alternativos, sendo uma premissa para o seu funcionamento correto. Outro protocolo relacionado é o protocolo SNMP, responsável por realizar coleta dos dados resultantes do protocolo spanning-tree e das informações contidas nas AFTs por meio das MIBs. No processo de atualização dos dados das AFTs, são utilizados os protocolos ARP e o ICMP, mantendo esses dados atualizados para que a coleta realizada pelo SNMP contenha as informações necessárias ao passo da descoberta de topologia com o algoritmo *Skeleton-tree*.

***Spanning Tree Protocol* (STP).** O STP é um protocolo L2 que determina melhores caminhos e apresenta a estrutura lógica de caminho único de árvore, sem ciclos, elegendo um dispositivo como raiz.

***Simple Network Management Protocol* (SNMP).** O SMNP é um protocolo que define padrões no uso do gerenciamento de dispositivos de rede, com o uso da Base de Informações Gerenciais (*Management Information Base - MIB*) (CASE et al., 1990).

Objetos gerenciáveis de uma rede são acessados através de um armazenamento de informações, denominado *Management Information Base* (MIB). Por sua vez, objetos MIB são definidos usando *Abstract Syntax Notation One* (ITU-T, 2002). Cada objeto tem um nome, uma sintaxe e uma codificação. O nome é um identificador de objeto (*Object ID - OID*) atribuído administrativamente, que especifica um tipo de objeto. O tipo de objeto junto com uma instância de objeto serve para identificar de forma exclusiva uma

instanciação específica do objeto (MCCLOGHRIE; ROSE, 1990). Na técnica proposta que usa por base o algoritmo *Skeleton-tree*, são usados apenas identificadores de objetos padrões no processo de coleta de dados, tornando-se independente de fabricantes.

O módulo *Bridge-MIB* é usado para gerenciar dispositivos compatíveis com IEEE 802.1d, o *spanning-tree*. O módulo define objetos para gerenciar dispositivos bridge MAC baseadas no padrão IEEE 802.1d, entre segmentos de rede local (LAN). As provisões são feitas para o suporte de *bridges* transparentes e para que esses objetos sejam aplicados às *bridges* interconectadas por sub-redes que não sejam segmentos da LAN (NORSETH; BELL, 2005).

A realização das coletas feitas na simulação de rede deste trabalho utiliza objetos desse módulo, podendo ser citado o grupo *dot1dStp*. Esse grupo contém as OIDs que indicam o funcionamento dos objetos referentes ao *spanning-tree*, armazenando informações de quantidade e estado das interfaces no STP, como endereços associados e demais informações relacionadas. Os objetos do grupo *dot1dStp* são definidos na RFC 4188 de maneira independente de fabricantes.

***Address Resolution Protocol* (ARP).** Neste trabalho, a utilização do protocolo ARP é proposta como alternativa ao protocolo ICMP (*Internet Control Message Protocol*) para envio de solicitações ativas na rede. O objetivo é provocar a atualização das tabelas de encaminhamento (AFTs) dos dispositivos intermediários na rede local e identificar dispositivos ativos. Observa-se que existe uma sobrecarga de protocolo maior no ICMP por trabalhar em uma camada acima no modelo OSI em relação ao ARP, agregando mais informações de endereçamento e controle no cabeçalho.

Na RFC 5227 (CHESHIRE, 2008) é especificado que uma solicitação ARP deve ser gerada quando uma interface é configurada, com o objetivo de descobrir se o endereço já está em uso. Teste adicional também deve ser executado em cada pacote ARP recebido, para facilitar a detecção passiva de conflitos em andamento. Entretanto, as interfaces consideradas neste estudo não possuem configuração diferenciada entre si, não havendo necessidade de abordar essas especificações.

***Internet Control Message Protocol* (ICMP).** O protocolo ICMP é parte integrante da implementação do protocolo IP. Mensagens ICMP são enviadas em várias situações, podendo ser mencionado quando um datagrama não pode alcançar seu destino, quando o *gateway* não tem a capacidade de armazenamento em *buffer* para encaminhar um datagrama e quando o gateway pode direcionar o host para enviar tráfego em uma rota mais curta (POSTEL, 1981). Ferramentas como *ping* e *traceroute* são amplamente difundidas e fazem uso do ICMP para realizar teste de conectividade entre dois dispositivos de rede. O uso do ICMP por meio do *ping* é originalmente utilizado como método ativo no passo de coleta de dados que ocorre anteriormente ao cálculo do algoritmo *Skeleton-tree*, realizando atualização dos dados necessários ao algoritmo para executar a descoberta de

topologia. Neste trabalho é reproduzido o algoritmo *Skeleton-tree* com uso originalmente do ICMP para atualização dos dados e confrontado com o uso exclusivo do ARP, na alteração proposta, para comparação dos resultados.

***Address Forwarding Table* (AFT).** AFT é uma tabela de mapeamento de endereços MAC com suas portas associadas em um dispositivo. Essa tabela é utilizada pelo switch para encaminhar os dados destinados a um determinado endereço MAC na porta física correta. A AFT aprende novos endereços MAC, elimina endereços MAC antigos e, em alguns dispositivos, permite exibir, adicionar, modificar e remover entradas AFT (endereços MAC na tabela).

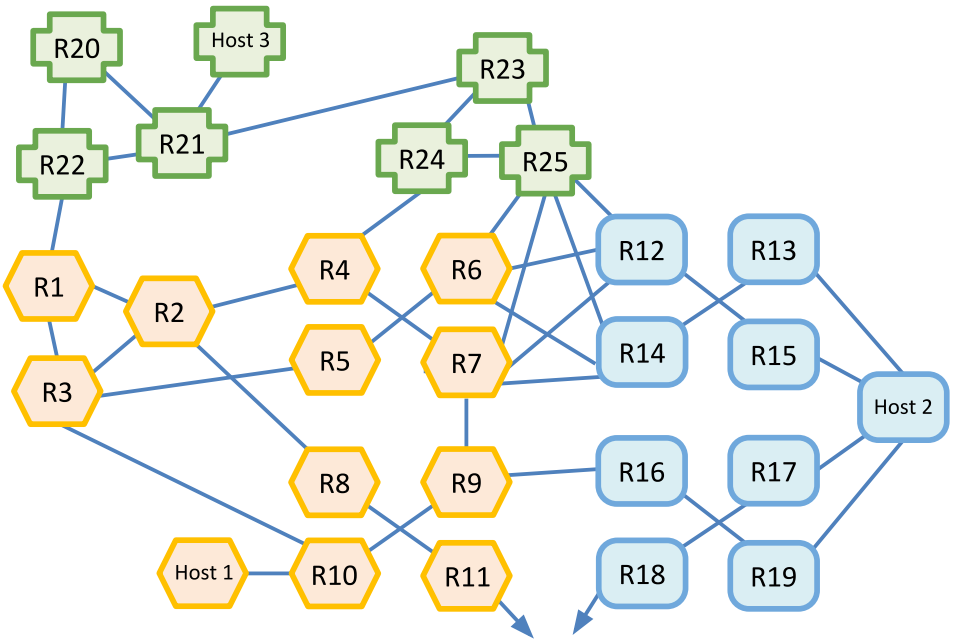
1.3 Níveis de Detalhamento de uma Topologia de Rede

O mapeamento da topologia pode ser dividido em quatro diferentes níveis de detalhamento: Nível-interface, Nível-roteador, Nível-PoP, Nível-AS.(MOTAMEDI; REJAIE; WILLINGER, 2015).

Nível-interface. A topologia com detalhamento em nível-interface (*interface-level*) adota o mapeamento 1-para-1 entre nós e IPs (MADHYASTHA et al., 2006), não tendo visibilidade dos dispositivos das camadas abaixo da camada de rede do modelo OSI (*hubs* e *switches*). O *traceroute* pode ser utilizado como ferramenta de medição no plano de dados do nível-interface, empregado na tarefa de descoberta. Nesse nível de detalhamento, os nós da topologia representam as interfaces lógicas entre os dispositivos, não havendo união de interfaces de um mesmo dispositivo. As ligações entre dispositivos representam a sua conectividade IP (L3). Neste trabalho é utilizado o detalhamento nível-interface como base, sendo o processo de descoberta realizado pelo algoritmo *Skeleton-tree*, incluindo elementos abaixo da camada de rede. Detalhes de como esse nível é utilizado na técnica são discutidos na Seção 3.4.

Nível-roteador. A topologia nível-roteador (*router-level*) apresenta os roteadores e as interconexões entre suas interfaces de rede. Nesse nível de detalhamento, os nós da topologia representam os dispositivos finais (*end-hosts*) com uma interface ou roteadores com múltiplas interfaces. As ligações entre eles representam a conectividade entre dispositivos de camada 3 do modelo OSI. O nível-roteador representa cada nó da topologia como o resultado da agregação de interfaces IPs que pertencem a um único roteador, conforme exemplo reproduzido na Figura 2. Caso existam conexões redundantes entre dois roteadores, suas interfaces são agrupadas e representadas como uma única conexão. As técnicas *Alias Resolution* e *Recursive Router Discovery* apresentadas em (GOVINDAN; TANGMUNARUNKIT, 2000), (SPRING et al., 2004), e (PANSIOT et al., 2010) são

Figura 2. Topologia representada com detalhamento nível-roteador.



Fonte: Adaptado de (MOTAMEDI; REJAIE; WILLINGER, 2015).

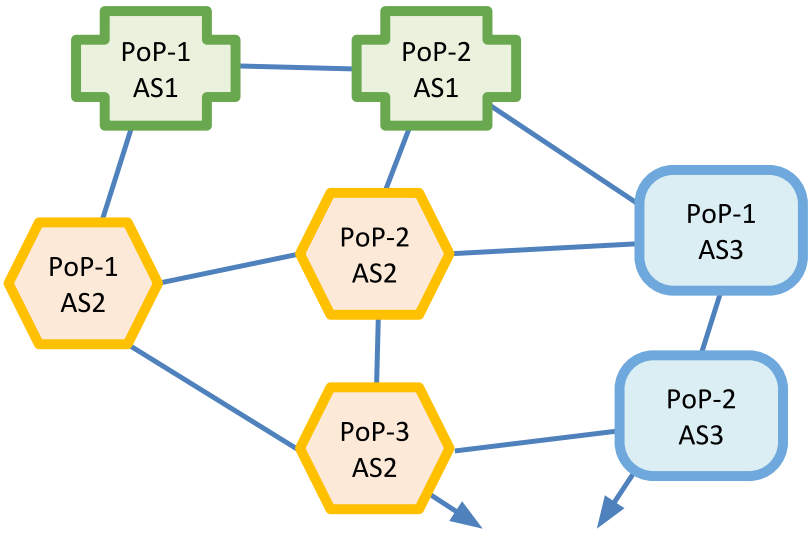
consideradas duas técnicas principais para a coleta de informações para a descoberta de topologia nível-roteador.

A técnica *Alias Resolution* consiste no processo de junção dos diversos endereços IPs de um roteador em um único nó representado, podendo esse processo ser derivado da topologia de nível-interface. A descoberta recursiva ou progressiva de rede (*Recursive Router Discovery*) é fundamentada na capacidade dos roteadores de redes armazenarem informações dos seus roteadores vizinhos.

A topologia de rede pode ser descoberta progressivamente com o uso as informações dos vizinhos. Em redes de roteadores habilitados com SNMP, uma lista de interfaces vizinhas pode ser identificada de uma entrada *MIB ipRoute Table* do roteador. Uma técnica semelhante de descoberta incremental é utilizada em (PANSIOT et al., 2010) com mensagens IGMP. Nessa técnica, quando uma mensagem *IGMP Ask Neighbors* é recebida, um roteador responde com um *IGMP Neighbors Reply*, no qual são listadas todas as interfaces do roteador vizinho.

Nível-PoP. A topologia de detalhamento nível-PoP tem o objetivo de identificar os locais nos quais um sistema autônomo (*Autonomous System* - AS) tem a possibilidade de trocar tráfego com seus vizinhos. Nesse nível de detalhamento, os nós da topologia representam a união dos dispositivos de um provedor em um ponto geográfico específico, para atender os clientes locais ou fazer interface com outros provedores. Normalmente abriga servidores, roteadores, *switches*, multiplexadores e outros equipamentos de interface de rede. Esses

Figura 3. Topologia representada com detalhamento nível-PoP.



equipamentos geralmente estão localizados em um Datacenter. A representação de um nó pode ser obtida pela junção de nós do nível-roteador que estejam em um mesmo ponto geográfico, desde que pertençam a um mesmo provedor. A topologia em detalhamento nível-PoP é apresentada na Figura 3. Dessa forma, os nós R1, R2 e R3 da Figura 2 são representados pelo nó PoP-1 AS2 na Figura 3, indicando que pertencem a um mesmo ponto geográfico. A topologia em detalhamento nível-PoP é útil para potenciais clientes que estejam interessados na cobertura geográfica do AS ou em saber os locais aos quais podem interconectar.

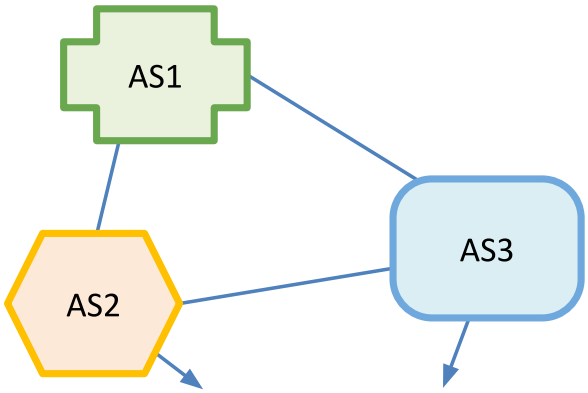
Nível-AS. O detalhamento mais abrangente de topologia da rede é o nível-AS. A topologia nesse nível é tipicamente modelada usando uma estrutura simples, no qual cada nó na topologia representa um AS identificado pelo seu número. O número do AS identifica uma coleção de prefixos IPs sob o controle de um único operador de rede. A representação de um nó nesse nível de detalhamento pode ser obtida com a união de todos os pontos de presença (PoP) dos prefixos IPS de um operador de rede na região. A visão da topologia em detalhamento nível-AS está representada na Figura 4.

1.4 Coleta e Medição

A visão precisa de uma topologia de rede obtida e a formação dos diversos níveis de detalhamento tem como passo inicial e a obtenção de dados obtidos dos dispositivos da rede para, então, ocorrer a análise destes dados.

A obtenção dos dados é realizada com o suporte de ferramentas de medição ou com

Figura 4. Topologia representada com detalhamento nível-AS.



a coleta de dados na própria rede, podendo ter origem no plano de dados ou no plano de controle (MOTAMEDI; REJAIE; WILLINGER, 2015).

As medições no plano de controle revelam informações tais como tabelas de roteamento, em contraste com as medições no plano de dados que têm como foco descobrir o caminho vigente que um pacote percorre. A ferramenta de medição mais simples no plano de dados é o *ping*, já mencionado também como método ativo. O *ping* que gera um pacote ICMP na rede e é normalmente usada para verificar se um dispositivo está ativo.

Ainda, é possível classificar o tipo de medição como passiva ou ativa. Na medição ativa são usados métodos que geram ativamente pacotes na rede para provocar a produção de dados que são atualizados e persistidos em tabelas nos dispositivos. Posteriormente estes dados são coletados pelas ferramentas de coleta. A medição é definida como passiva quando os dados necessários já estão disponíveis, geralmente armazenados nos dispositivos de rede, disponíveis para serem coletados.

Destacam-se como exemplos de ferramentas para medição ativa e passiva o já mencionado *traceroute* e o BGP *monitor* respectivamente. BGP *monitors* são responsáveis por coletar informações de rotas BGP através de um conjunto de coletores. Um *monitor* (coletor) representa um roteador com informações do BGP. O SNMP, já mencionado, também é considerado um exemplo de medição passiva.

1.5 **GNS3**

O *Graphical Network Simulator 3* (GNS3) é uma ferramenta, gratuita, destinada a criar, projetar e testar redes de computadores de forma visual em um ambiente virtual controlado de funcionamento contínuo (não discreto). Seu uso pode ser de grande utilidade para estudos realistas da operação de dispositivos de rede ou mesmo construindo uma rede

de telecomunicações de longa distância.

O GNS3 oferece uma maneira fácil e prática de projetar e construir redes de variados tamanhos sem a necessidade de hardware (e de seu custo agregado). Está disponível para Windows, MAC e Linux. No GNS3 a rede pode ser visualmente criada de forma intuitiva por meio da interface gráfica, representando os nós através de dispositivos virtuais e fazendo a interconexão das suas interfaces para criarem os enlaces.

Simulações de rede em tempo real, de forma não discreta levando os tempos de espera dos protocolos, podem ser realizadas no GNS3 para análise de funcionamento e testes de pré-implantação. Ele executa o sistema operacional que emula o comportamento real do hardware de rede. Atualmente é possível testar roteadores, *multilayer switches*, *firewalls* de mais de 20 fabricantes de dispositivos de rede diferentes, em ambientes virtuais livres de riscos.

É possível criar dinamicamente topologias de rede para testes de solução de problemas e prova de conceito (*proof of concept - POC*). Também é possível testar redes antes de criá-las fisicamente, reduzindo o tempo necessário para colocar uma rede de produção em funcionamento por meio da preparação antecipada da configuração dos dispositivos.

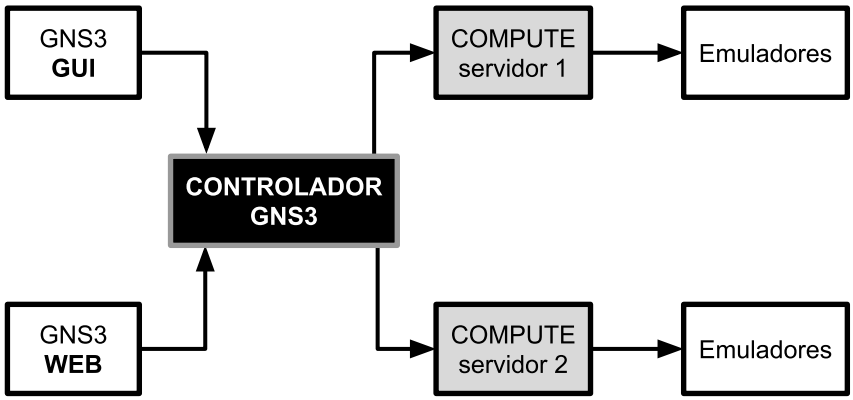
Topologias e laboratórios personalizados podem ser criados para treinamento de profissionais de telecomunicações que buscam certificações na área e estudos de caso, mostrando-se eficiente em implementar um ambiente virtual com mesmos recursos de uma rede real em produção. Dentre os recursos capazes de serem emulados, existe suporte para as tecnologias de *Etherchannel*, PVST+, RPVST+, MST, *Port Security* dentre outras de camada 2 do modelo OSI.

A arquitetura do GNS3 pode ser dividida em interface do usuário (*Graphical User Interface* - GUI), o controlador, o compute e os emuladores.

O controlador orquestra os demais elementos e gerencia o estado de cada projeto. Apenas um controlador deve ser executado. A GUI exibe a topologia que representa um projeto em uma tela e permite executar ações no determinado projeto, enviando solicitações de API ao controlador. O *compute* controla os emuladores para executar os nós. Geralmente é iniciada uma instância de emulador para cada nó. O esquema desse funcionamento pode ser observado na Figura 5. Toda a comunicação entre os recursos do GNS3 é feita via HTTP usando o formato JSON (JavaScript Object Notation). Esse é um formato de arquivo de padrão aberto que usa texto legível para transmitir objetos de dados em pares de atributo e valor.

O GNS3 pode emular de forma nativa alguns dispositivos simples, como *switches* não gerenciáveis e VPCs (Virtual PCs - PCs virtuais). É possível também a inclusão de novos dispositivos com o uso de *appliances*. Os *appliances* são módulos usados para representar e acrescentar dispositivos emulados predefinidos à rede. Os *appliances* poder ser compartilhados, importados e inseridos na topologia como dispositivos virtualizados prontos para uso. Os desenvolvedores do GNS3 disponibilizam também diversos *appli-*

Figura 5. Esquema simplificado da arquitetura do GNS3.



Fonte: Adaptado de (GNS3, 2018).

ances preparados para serem importados no GNS3, a exemplo de *switches*, roteadores, *firewalls* e servidores de diversos fabricantes, entre outros recursos de rede.

A criação manual da topologia no GNS3 é realizada com procedimentos de arrastar-e-soltar os appliances disponíveis na área de trabalho do simulador, na qual posteriormente os dispositivos são interconectados por meio de suas interfaces disponíveis. A topologia é concluída com a ligação e configuração dos dispositivos com endereçamento das sub-redes envolvidas.

A abordagem adotada para avaliação da técnica para a solução obtida foi utilizar o GNS3 para emular estruturas de redes locais, operando e executando a pilha de protocolos IP e ferramentas como o SNMP, de onde são coletados os dados de entrada para a descoberta da topologia com o algoritmo *Skeleton-tree*.

A utilização do GNS3 não é citada por Bejerano nem o ambiente de testes em que foram executadas suas simulações do *Skeleton-tree*. A escolha pelo GNS3 é feita com base na sua característica de simular a rede de forma virtualizada e contínua sem os custos envolvidos na aquisição do hardware que seria necessário. Dessa forma a topologia criada no simulador apresenta as funções reais dos equipamentos utilizados, podendo ser conectados a uma rede não simulada (rede física).

2 TRABALHOS RELACIONADOS

O artigo *A Survey of Techniques for Internet Topology Discovery* (MOTAMEDI; REJAIE; WILLINGER, 2015) faz uma comparação de diversas técnicas de descoberta de topologias de redes na *Internet*, caracterizando os seus níveis de detalhamento, incluindo o explorado neste trabalho. O artigo aborda a medição e modelagem de mais de 100 trabalhos científicos de um período de 20 anos, nos níveis de detalhamento interface, roteador, PoP e AS, introduzidos na Seção 1.3. O artigo conclui que usar mais medições do mesmo ciclo de *tracerouters* ou mesmo conjunto de *BGP monitors* não resolve a incompletude das topologias de rede. Nesse artigo é proposto uma lista de perguntas que devem ser feitas antecipadamente ao início do processo de descoberta de topologias de redes, reproduzidas a seguir:

- Quais conjuntos de dados são usados ou gerados para o trabalho planejado?
- Quais técnicas foram usadas para obter os dados?
- Quais são as limitações (conhecidas) das técnicas usadas e o que se sabe sobre como essas limitações afetam a qualidade dos dados?
- Como os problemas conhecidos da qualidade de dados podem afetar os resultados do trabalho planejado?
- Se os problemas da qualidade de dados conhecidos forem considerados secundários, os resultados e as descobertas obtidas resistirão a um exame mais detalhado com base em dados alternativos ou domínio de conhecimento disponível?

O objetivo dessas perguntas é organizar a verificação das limitações nas técnicas de descoberta de topologias usadas e como elas podem afetar os resultados. Dessa forma é possível identificar quais as ferramentas, conjuntos de dados e técnicas usadas são mais adequados para descobrir a topologia com êxito no cenário desejado (MOTAMEDI; REJAIE; WILLINGER, 2015).

No artigo *Topology Discovery Using Cisco Discovery Protocol* (RODRIGUEZ, 2009) é descrita uma técnica baseada em CDP (*Cisco Discovery Protocol*) para descoberta de topologias de redes. Essa técnica é capaz de descobrir a topologia física da rede, incluindo enlaces que estejam em estado *Forwarding Block* no *spanning-tree*. Nesse estado a interface de um *switch* não encaminha dados e atua como se fosse um enlace desconectado, transmitindo apenas os quadros referentes ao funcionamento do *spanning-tree*. O uso do CDP fornece informações detalhadas sobre a conectividade dos dispositivos, sendo apresentado como uma técnica útil para pequenas redes baseadas em dispositivos Cisco ou compatíveis com esse protocolo. O CDP é um protocolo proprietário que mantém

informação sobre estado da rede no dispositivo, podendo funcionar em servidores, roteadores, *bridges* e *switches*. Esses dados podem ser extraídos do objeto CISCO-CDP-MIB por meio do SNMP. Seu funcionamento é fundamentado no envio e recebimento de mensagens *multicast* periódicas para determinar seu estado e dos dispositivos vizinhos. Objetos independentes do fornecedor Cisco estão padronizados na RFC 2922 (BIERMAN; JONES, 2000).

O algoritmo de descoberta concebido por Rodriguez (RODRIGUEZ, 2009) é iniciado com a seleção de um dispositivo IP da rede a ser descoberta. Esse dispositivo é tomado como dispositivo raiz, servindo de origem para leitura posterior dos IPs de dispositivos vizinhos identificados na MIB. Os IPs descobertos são colocados em fila e processados até que a fila esteja vazia. O dispositivo raiz é considerado o nível 1 de descoberta, enquanto dispositivos interconectados a ele são de nível 2 e assim por diante. A descoberta da topologia ativa e todas as conexões entre os dispositivos é apresentada como resultado final, incluindo os enlaces bloqueados pelo *Spanning Tree Protocol*. Enlaces interconectados às interfaces que tenham sido bloqueados administrativamente não podem ser descobertas. Essa técnica se apresenta viável apenas em cenário de topologia composta exclusivamente por dispositivos Cisco, devido ao uso do protocolo CDP proprietário, não sendo eficiente na detecção de elementos não gerenciáveis e de outros fabricantes.

O estudo apresentado no artigo *Topology Discovery in PROFINET* (SCHAFER; FELSER, 2007) aborda a descoberta de topologia em redes de automação baseadas em *Ethernet*, denominadas redes PROFINET. A descoberta automática da topologia física é desenvolvida com o protocolo de descoberta de camada de enlace (*Link Layer Discovery Protocol* - LLDP) (CONGDON, 2002). O trabalho realiza a análise de diferentes implementações e apresenta vários problemas do uso do LLDP, estendendo à interação com sistema de gerenciamento de rede. O sistema de descoberta é composto por um *IP-controller* (controlador-IP), um ou mais dispositivos de entrada e saída (*In Out* - IO) e um possível supervisor de IO. Os dados sobre a topologia de rede do protocolo LLDP são armazenados em cada dispositivo e a coleta desses dados é possível com o uso do SNMP, para construir a topologia.

Nessa técnica, a descoberta da topologia é iniciada com a identificação das conexões físicas, por meio de ferramentas e algoritmos baseados em L2. Isso é possível com o uso de *Switches* gerenciáveis com suporte ao SNMP, que armazenam dados da tabela de encaminhamento (*Address Forwarding Table* - AFT) em local específico definido pela MIB. O LLDP padrão 802.1ab é outra tecnologia L2 e sucessor ao CDP, independente de fabricante. O LLDP tem função similar ao CDP, mas não compatível e está presente em diversos switches gerenciáveis. Dispositivos compatíveis com LLDP enviam e recebem *LLDP Data Units* (LLDPDU) em todas as interfaces físicas, armazenando os dados na LLDP-MIB local. O *Media Service Access Point* (MSAP) consiste na identificação de dispositivo e porta de forma única na rede observada. Quando um dispositivo recebe um

pacote formado pelos *Type-Length-Value-blocks* (TLV) com informações do remetente, associa na LLDP-MIB o MSAP com o número da porta local de entrada. A estação de gerenciamento de rede (*Network Management Station* - NMS) realiza troca periódica de LLDPDU para manter vizinhos atualizados, coletar os dados por meio do SNMP e os combina, resultando em um banco de dados contendo todos os dispositivos e suas conexões.

No artigo mencionado de Schafer, a descoberta de topologia é confrontada com alguns limites e problemas, como switches sem suporte a LLDP, interpretações diferentes dos pacotes e do uso do SNMP. Casos em que existam na topologia *switches* executando RSTP (*Rapid Spanning Tree Protocol, IEEE 802.1w*), cada switch deve estar ciente dos endereços do LLDP para encaminhar corretamente em outros segmentos. Pacotes são encaminhados pelo switch em todas as portas caso não estiver ciente desses endereços, causando formação indevida de vizinhança. A conclusão do artigo expõe que a técnica é extensível via TLV opcional ou específico da organização, tornando possível resolver problemas além da descoberta de topologia. Entretanto, a técnica Skeleton-tree utilizada neste trabalho não apresenta tal dependência do LLDP.

No artigo *Topology Discovery of Hierarchical Ethernet LANs without SNMP support* (NOWICKI; MALINOWSKI, 2015) é apresentado técnica que não requer acesso administrativo aos dispositivos intermediários para coleta de dados. No artigo é descrito técnica com base no *Ethernet* e que não necessita instalar agentes nos dispositivos para realizar a descoberta de topologia hierárquica de rede. Também não são necessárias alterações de *hardware*, *firmware* ou sistema operacional. A proposta da técnica consiste em instalar um processo, denominado *daemon*, em segundo plano nos nós finais. Esse processo interage com as solicitações quando necessário. Os *daemons* são definidos como *master daemon* (MD) e *slave daemon* (SD), funcionando como mestre e escravo respectivamente. Dessa forma, injeta pacotes *Ethernet* com endereços MACs *unicast* não atribuídos pela IANA. A técnica assume não haver filtragem de MAC nem *hubs* na rede, sendo uma ferramenta sugerida por Nowicki para diagnóstico de rede, gerenciamento de recursos e implementação de servidor.

Na primeira fase da técnica, o MD descobre todos os SD. Nessa fase inicial de descoberta o dispositivo *master* envia mensagens para todos os *slaves* e registra seus endereços MACs por meio das respostas. Na segunda fase, a descoberta dos *switches* diretamente conectados é realizada. Nessa fase é apresentado um algoritmo que faz uso de 2 endereços *unicasts* não atribuídos pela IANA. O *master* intermedia essa comunicação para identificar os *switches* diretamente conectados. Na terceira e última fase, a descoberta dos *switches* não diretamente conectados é feita. Esses *switches* são identificados por meio de isolamento de ilhas de switches, através da análise em grupos de 3 e permutação com novos dispositivos para formar a hierarquia.

A técnica considera estrutura de árvore tendo o *gateway* como raiz e a desco-

berta iniciando no nível mais baixo, oposto ao raiz. Na descoberta das áreas de *switches* agrupados na fase 2, *switches* de hierarquia imediatamente superior são detectados e é determinado o grau da topologia. A topologia correta é apresentada ao final, sem necessidade de alterar *switches* ou executar coletas. A técnica, porém, não aborda topologias de complexidade maior em relação a quantidade de dispositivos de rede. A criação de um *kit* de descoberta de rede também é proposto, com plataformas de hardware de baixo custo. A quantidade de *daemons* necessários para a descoberta de rede está relacionada com a quantidade existente de dispositivos na rede. A técnica não apresenta uma solução centralizada, dependendo da implementação distribuída de *daemons*. A implementação não foi abordada em redes de grande porte.

Tracing of Hardware Addresses in Layer Two Bridged Networks (SCHOLZ; KRUSE, 2015) é um artigo de estudo baseado em *tracemac* e uma proposta de extensão para LLDP, com TLV (*Type Length Value*) personalizado. O *tracemac* é considerado o rastreamento de dispositivos da camada de enlace (*trace* L2) em analogia ao rastreamento de dispositivos da camada de rede (*trace* L3). No estudo é abordado o formato e uso do TLV em 3 cenários para dispositivos com diferentes características. O primeiro cenário é quando não é encontrado registro para o dispositivo (*No Entry Found*). No segundo cenário o registro existente é local (*Entry is Local*). O terceiro cenário é quando o registro existente não é local (*Entry is not Local*).

Na fase de inicialização da técnica com o TLV personalizado é utilizado o protocolo de resolução de endereço reverso (*Reverse Address Resolution Protocol* - RARP) para atualizar tabelas, seguidos de novos RARPs e *LLDPDUS* (*LLDP Data Units*) que ocorrem nas devidas VLANs. A fase posterior executa o rastreamento. Caso não exista correspondência MAC, ou seja, *No Entry Found*, o *tracemac* é interrompido. Caso exista correspondência MAC ocorre o envio de LLDPDU contendo TLVs *Chassis ID*, *Port ID* e *Time-to-Live* para rastrear o dispositivo. Nesse processo a VLAN é identificada no recebimento da resposta de cada *Trace Request TLV* personalizado.

No cenário da técnica com TLV personalizado em que há registro local, a resposta LLDPDU não é *multicast* e o TTL TLV (*Time to Live* do TLV) é definido com valor igual a zero. No cenário em que o registro não é local, a resposta LLDPDU ocorre em formato *multicast* e o TTL é definido de acordo com o padrão 802.1ab. Na técnica, o *Trace Reply TLV* deve ser o primeiro na sequência de possíveis TLVs e ter um tempo limite para o caso de haver dispositivo incompatível. Caso os *LLDPDUs* sejam encaminhados de forma transparente, apenas um dispositivo apresenta mapeamento correto. Caso a resposta de fim do trace estiver sem a informação de porta de saída, o trace não é abortado e passa a aguardar o *timeout* ou a possível chegada respostas vindas de outros dispositivos. O uso de TLV identificador é necessário no caso de rastreios simultâneos, para garantir que a resposta seja feita a uma solicitação específica. Uma mensagem de notificação pode ser disparada durante o processo de rastreamento, contendo as informações de identificação

do dispositivo e interface.

O estudo apresenta ainda a elaboração de máquinas de estado para enviar e receber LLDP, controlar o tempo limite, enfileiramento, propagação e comunicação. Os resultados têm como base uma rede implementada contendo enlaces WAN, 100 *bridges* e topologia em malha. A rede é considerada convergida e a criação de ferramenta (*plugin*) para o analisador de protocolos *Wireshark*. O foco da coleta de informações ocorre sobre a topologia física, sendo o protocolo ciente de VLAN e podendo atuar em enlaces ponto-a-ponto ou compartilhados. A técnica também extensível com TLVs adicionais, com possibilidade de automatização para solução de problemas, documentação e verificação. Observa-se que os dispositivos de camada 3 não encaminham *LLDPDUs* e os dispositivos não cientes do *tracemac* ignoram os TLVs desconhecidos, causando a interrupção do *trace*.

O algoritmo *Skeleton-tree* é apresentado no artigo *Taking the Skeletons Out of the Closets* (BEJERANO, 2009), no qual é proposto uma técnica para descobrir a topologia física de LANs *Ethernet* heterogêneas de grande porte. Essas LANs podem incluir várias sub-redes, bem como dispositivos de rede não cooperativos. Um dispositivo definido como não cooperativo é aquele que não possui endereço IP e no qual não é possível ser feita coleta de dados, a exemplo dos *hubs* e *switches* não gerenciáveis. O esquema utiliza apenas informações de MIBs genéricas e não requer nenhuma modificação de hardware ou software dos elementos de rede. A técnica pode inferir a topologia de rede por meio das análises dos dados coletados dos dispositivos de rede. Os resultados das simulações mostram que o esquema infere a topologia de forma centralizada e sem alterações na estrutura de rede. Por ser esse a técnica definida como referência para este trabalho, o algoritmo *Skeleton-tree* é apresentado mais detalhadamente na Seção 3.1.

As diversas técnicas apresentadas de descoberta de rede exploram o nível de detalhamento da topologia almejada, com foco na precisão da topologia resultante que representa a rede. Nas técnicas pouco é mencionado do tempo de execução dos métodos expostos e dos impactos possíveis da variação do tempo limite de espera no processo de atualização e coleta dos dados necessários. A implementação da técnica *Skeleton-tree* consolida os resultados de Bejerano, explorando a melhoria no tempo total de descoberta da topologia.

3 *SKELETON-TREE* MODIFICADO

Neste capítulo a técnica *Skeleton-tree* [2] original utilizada na descoberta é apresentada com mais detalhes e são discutidas as soluções adotadas, bem como as restrições existentes. Na sequência é discutida a implementação desenvolvida em *Python* com base na técnica original, permitindo uma avaliação do algoritmo. Ao processo original é proposta, também, a introdução do ARP como alternativa ao ICMP, para acelerar o processo de coleta dos dados usados na descoberta da topologia.

3.1 A técnica Skeleton-tree

A Figura 6 apresenta a sequência completa de passos da técnica *Skeleton-tree* para descoberta de topologia:

- **Atualização das AFTs (ICMP ou ARP).** Inicialmente é realizado um procedimento para provocar a atualização das AFT e convergência do *spanning-tree* é executado para que o algoritmo trabalhe com dados suficientes para montar a topologia correta (Seção 3.2). O método original utiliza ICMP como forma de atualizar as AFTs. Neste trabalho é proposto o uso exclusivo do ARP como método ativo (Seção 3.7);

- **Coleta por SNMP.** Em seguida os dados das tabelas AFT são coletados utilizando SNMP por meio de MIB padrão (BRIDGE-MIB), sem dependência de fabricante (Seção 3.3);

A partir deste ponto o algoritmo é executado em duas etapas (Seção 3.6):

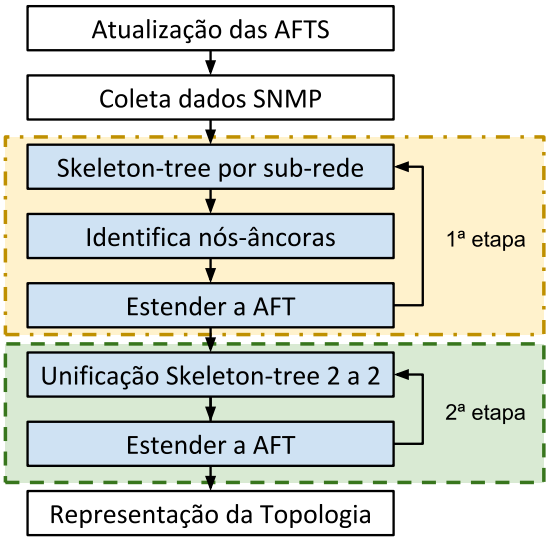
1ª Etapa. Sub-árvores.

- **Criação da estrutura de árvore *Skeleton-tree*.** Para cada sub-rede é criada uma estrutura de dados em árvore, com base no spanning-tree, e que depois será consolidada em uma estrutura única;

- **Identificação de nós âncoras de cada árvore *Skeleton-tree*.** Uma parte importante do algoritmo é a identificação de nós, chamados âncora, que depois são

[2] Cabe observar que *Skeleton-tree* é utilizado nos artigos de referência como o nome da técnica, nome do algoritmo e da estrutura de dados em árvore utilizada para representar os estados intermediários e a topologia final. Ao longo do texto buscou-se identificar que instância estava sendo referida.

Figura 6. Passos da técnica *Skeleton-tree*



utilizados para consolidar as estruturas de dados das árvores;

- **Estender a AFT para cada árvore *Skeleton-tree*.** Dados das AFTs dos nós âncoras são combinados para proporcionar uma tabela AFT mais completa para a etapa de unificação;

2ª etapa: Unificação.

- **Unificação da *Skeleton-tree*.** Duas a duas, cada árvore *Skeleton-tree* é consolidada em uma nova Skeleton-tree até que não exista mais Skeleton-tree ou que as árvores não possam mais ser unidas;

- **Estender a AFT para cada árvore *Skeleton-tree* unificada.** Dados das AFTs dos nós âncoras são combinados para proporcionar uma tabela mais completa;

- **Geração da representação da topologia.** Topologia descoberta ou falha na unificação.

O algoritmo *Skeleton-tree* é capaz de realizar a descoberta de topologia no nível-interface, identificando os dispositivos com interfaces IPs e indo além, incluindo os dispositivos da camada 2 e 1 (L2 e L1) do modelo OSI, por meio da análise dos dados coletados. Tem a habilidade de descoberta de dispositivos em sub-rede heterogênea, podendo ser mencionados dispositivos roteadores, *switches*, *switches* não gerenciáveis, *host* e *hubs*.

Por conta dessa capacidade de identificar dispositivos L2 e L1, a descoberta de topologia é realizada na sub-rede local e, assim, a técnica se restringe a um único domínio de difusão, usando mensagens L2 específicas de difusão (*broadcast*) como primeiro passo

na identificação dos nós da topologia. Por este motivo também, admite-se que os nós fazem parte de uma mesma VLAN. As sub-redes pertencentes ao domínio de difusão são definidas por seus endereços lógicos de rede (IPs). Em consequência, os endereços MAC coletados para o algoritmo construir a topologia fazem parte desse mesmo domínio de difusão (rede L2), e figuram na mesma árvore de *spanning-tree*, independentemente de estarem em redes lógicas diferentes (rede L3).

Entre os dispositivos representados podem existir nós-internos e nós-folhas. Um nó-interno é o nó que faz o encaminhamento dos dados internamente na rede, de um nó de origem à um nó de destino, sendo representado neste trabalho pelo *switch*. Um nó-interno pode participar da formação de topologia de mais de uma sub-rede local, desde que faça parte de algum dos caminhos formados entre dois nós-folhas da rede em análise. Os nós restantes da topologia são denominados nós-folhas. O nó-folha é representado aqui pelo *host* e a estação NMS. Desta forma, os nós v_1 ao v_6 da topologia de referência apresentada na Figura 7a são nós-internos, enquanto os nós a ao m e x ao z são nós-folhas.

O conjunto de dispositivos pertencentes a uma sub-rede é caracterizado por possuir nós exclusivos à sub-rede (nós-folhas), definidos por seu endereço IP e contendo um MAC único. No caso da estação NMS, as interfaces para cada sub-rede são identificadas como nós-folhas distintos. Entretanto, os nós-folhas que representam o NMS são excluídos do cálculo do algoritmo, não influenciando na descoberta das topologias.

A técnica de descoberta de topologia utiliza a árvore *spanning-tree*, modelada na forma de uma árvore não direcionada, caracterizada por representar todos os nós-folhas e nós-internos do domínio de difusão, independente da sub-rede lógica (IP) que pertencem. O conjunto de dispositivos da árvore não direcionada é diferente do conjunto de dispositivos das sub-redes. Nesta árvore não direcionada, todos os dispositivos são representados, incluindo *switches* não gerenciáveis e *hubs*. No entanto, o conjunto dos dispositivos da sub-rede é representado apenas pelos dispositivos com endereçamento IP pertencentes a ela. A formação da árvore não direcionada é o objetivo da técnica de descoberta, representando a rede explorada. Porém os nós-internos não contém os dados dos nós não gerenciáveis e *hubs* que se pretende descobrir. Um exemplo da diferença entre a árvore não direcionada e as árvores das sub-redes é observado na topologia de referência apresentada na Figura 7.

A técnica utilizada identifica as interfaces L2 por meio das entradas AFTs para fazer a identificação dos nós-folhas. A varredura para a identificação de nós-folhas e coleta SNMP é realizada por sub-rede, sendo criada assim cada árvore de sub-rede. Nesse processo, a porta do nó-interno contendo entrada AFT (MAC) do nó-folha é associada a sub-rede envolvida por meio do IP desse nó-folha.

3.2 Atualização das AFTs

As informações contidas nas AFTs são dinamicamente obtidas em tempo de operação dos *switches*, na medida em que os dispositivos da rede precisam se comunicar. Assim, para a técnica de descoberta é necessário garantir que, no momento da coleta de dados, as AFTs distribuídas nos *switches* da topologia que se quer descobrir, vão estar, no conjunto, completas. Ou seja, vão conter os endereços MAC de todos os dispositivos da rede.

Observa-se que os endereços MACs aprendidos de forma dinâmica são removidos depois de um tempo determinado ou sobrepostos por endereços que realizaram comunicação mais recente. Dessa forma, não é possível garantir o conhecimento de todos os dispositivos da rede antecipadamente sem que ocorra a atualização dos dados dos seus antes do processo de coleta. Por isso, é importante garantir que as AFTs estejam completas e atualizadas. A atualização das AFTs na técnica original do algoritmo é provocada com a utilização do ICMP, forçando, a partir da estação de gerência, de forma ativa, a comunicação com cada dispositivo, em todas as sub-redes coexistindo na mesma rede física, e, assim, a consequente atualização dos endereços L2 nas tabelas envolvidas.

Os *switches* não utilizam as informações L3, o conteúdo (IP), em sua operação. O encaminhando dos pacotes é realizado com base no seu endereço MAC (L2) de destino. Assim, os pacotes ICMP (L3) são encaminhados por um *switch* com base no seu endereço MAC (L2). Por outro lado, uma solicitação ARP (*ARP request*), L2, é enviada de um dispositivo para qualquer outro nesta rede, por difusão. Uma solicitação ICMP *echo* (*Echo request*) em difusão só é recebida pelos dispositivos em uma mesma rede IP. Em consequência de as sub-redes coexistirem na mesma estrutura física, todas os nós recebem a solicitação ARP e não apenas os de uma sub-rede. Dessa forma, no caso em que o endereço físico (MAC) não estiver presente no cache (tabela ARP), a comunicação direta com ARP elimina a necessidade do ICMP ter que esperar resolver tal endereço físico (MAC) para depois enviar o *ping* para o endereço lógico (IP).

Cada pacote transferido pelo *switch* resulta na atualização do endereço MAC de origem e destino na tabela de encaminhamento, utilizada posteriormente no processo de coleta para o cálculo do algoritmo.

Caso uma solicitação ICMP seja feita para um endereço destino não presente na tabela de encaminhamento (AFT), o processo ICMP é suspenso e então iniciado o processo de descoberta por meio do protocolo ARP. O ARP busca descobrir o endereço físico do destino desejado por meio de mensagens de difusão L2. Somente após a conclusão do processo de descoberta do ARP e atualização da tabela de encaminhamento, o processo ICMP é retomado para tentar alcançar o endereço lógico de destino. Essa característica de comportamento é intrínseca aos protocolos mencionados

Observando essa característica é possível notar a vantagem de utilizar diretamente o ARP com endereço de difusão, eliminando o processamento extra e possíveis falhas

do uso do ICMP. É então criado e enviado um pacote ARP em difusão para a sub-rede IP, provocando a resposta simultânea de todos os dispositivos com IPs contidos nesse segmento. A resposta ao ARP também ocorre em caso de dispositivos em que a resposta ao ICMP foi bloqueada por medida de segurança. Desta forma garantindo resultado mais genérico e completo que o ICMP para topologias mais complexas.

O cabeçalho do *Ethernet* é formado tendo como endereço de destino o endereço L2 de difusão e o cabeçalho do ARP utiliza como destino o endereço da rede a ser analisada. Assim, os pacotes gerados chegam simultaneamente a todos os dispositivos do domínio de difusão e apenas as respostas da sub-rede IP analisada são capturados, identificando então os nós ativos.

A identificação dos nós existentes na rede por meio do método ativo de envido de pacotes, ICMP ou ARP, resulta também na atualização das entradas nas tabelas AFTs para posterior coleta com o SNMP. O método ativo se faz necessário pois a escuta promíscua poderia não resultar em dados atualizados. Dispositivos ociosos, que não transmitem dados continuamente, não seriam detectados pois suas entradas na AFT teriam sido removidas por tempo de inatividade.

3.3 Coleta de Dados SNMP

Os dados necessários para a execução do algoritmo são obtidos pela coleta das informações das tabelas de encaminhamento, AFTs, dos *switches* gerenciáveis que compõem a topologia descoberta. A coleta é feita por SNMP, que utilizada apenas MIBs genéricas para acessar os objetos necessários.

A estratégia para coleta de dados adotada tem alguns pontos que devem ser ressaltados. Relembrando, a atualização das AFTs e coleta de dados é realizada a partir do *host* especial, a estação de gerência (NMS). Esse *host* está conectado diretamente à mesma rede cuja topologia quer se descobrir. Considerando uma rede TCP/IP, o NMS conhece necessariamente o endereço IP e a máscara da rede ao qual está conectado (ou endereços IPs, no caso de ter acesso à várias sub-redes).

Assim, como primeiro passo, a estação de gerência utiliza o *ping* como ferramenta para ativamente fazer acesso à cada elemento ativo da rede (ou de cada sub-rede à qual está conectado), uma varredura. Com as mensagens de ICMP-echo e ICMP-response, envolvendo todos *hosts* e equipamentos gerenciáveis (que têm um número IP e respondem à mensagens ICMP-echo) ativos, espera-se provocar a atualização das tabelas AFT, tendo convergido o *spanning-tree* na rede, para que a coleta obtenha dados completos, mesmo se a rede estiver completamente ociosa. Em seguida, o *host* NMS obtém, de cada elemento que possua um agente SNMP — geralmente os roteadores e *switches* gerenciáveis (ou seja, que têm um número IP) — as tabelas de AFT, contendo registros de endereços MAC e

sua associação com as portas de cada equipamento que respondeu à consulta. A sutileza está em se utilizar o L3, para se obter informações de L2. Não se coletam tabelas de rotas, por exemplo. As informações de L2, e não as de L3 é que serão utilizadas pelo algoritmo *Skeleton-tree* para remontar a topologia da rede e identificar a existência de elementos L2 e L1 que não respondem à ICMP-echo, como *switches* não gerenciáveis e *hubs*.

A técnica também leva em consideração que o protocolo *spanning-tree* está em execução por se basear na sua estrutura de árvore e para identificar portas ativas.

A coleta dos dados da AFT ocorre por meio do SNMP através da MIB de *bridge* (BRIDGE-MIB), que contém as informações apenas das interfaces ativas do dispositivo, evitando assim que interfaces não conectadas participem do processo de descoberta. Com essa MIB é possível identificar os MAC associados a cada porta, indicando o caminho único para chegar ao destino. Os dispositivos de rede são assumidos em um mesmo domínio de difusão, conforme já mencionado, contendo endereços lógicos (IPs) em redes diferentes e endereços físicos (MACs) no mesmo domínio.

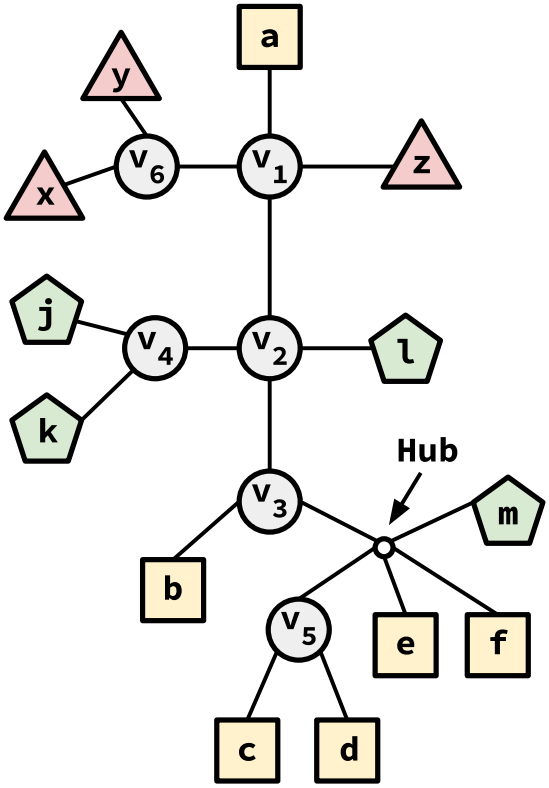
Na prática, existe a possibilidade do processo de preenchimento das AFTs não produzir as informações necessárias, devido ao tamanho limitado das tabelas que dão suporte às AFTs nos equipamentos e ao processo de descarte de registo por tempo ocioso. Esse problema também é mencionado por Bejerano e contornado com o envio de mensagens apenas para um conjunto limitado de nós por vez. As informações necessárias das AFTs são obtidas por meio da união de todas as coletas realizadas. Testes preliminares com a técnica *Skeleton-tree* reproduzida sem a coleta particionada não apresentaram resultados positivos suficientes na descoberta das topologias. Quando o tempo limite de espera é reduzido, as simulações também não apresentam resultados eficientes mesmo com o particionamento e união das coletas, ocorrendo falhas no processo de descoberta da topologia, conforme pode ser visto no Capítulo 5.

3.4 **Ferramentas e Nível de Detalhamento**

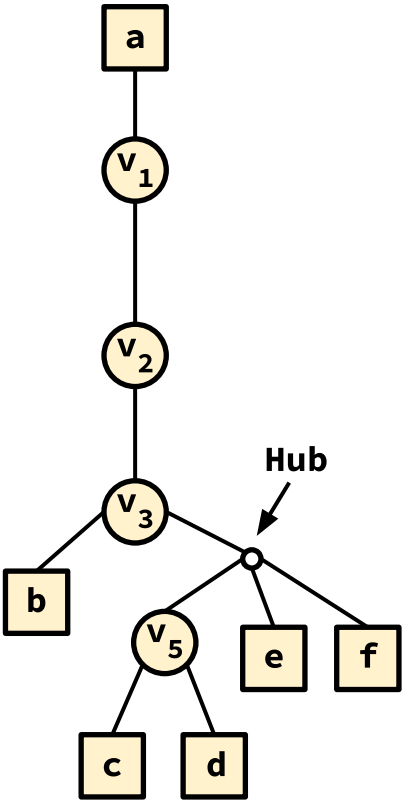
Nesta seção apresentamos uma discussão sobre as ferramentas empregadas na técnica *Skeleton-tree* e como o detalhamento de nível-interface é tratado neste trabalho. Um aspecto importante de nosso estudo da técnica de descoberta foi entender a diferença quando são considerados os dispositivos abaixo de L3 e quando não são considerados.

O *traceroute* é a ferramenta mais usada para mapear a topologia da rede com detalhamento nível-interface (MOTAMEDI; REJAIE; WILLINGER, 2015). Apesar disso, o *traceroute* não é capaz de realizar a identificação e mapeamento de dispositivos abaixo da camada 3 do modelo OSI, como *switches* e *hubs*. Isso ocorre devido à sua característica de trabalhar na camada 3. O *traceroute* usa a identificação de endereços lógicos (IPs) como uma das condições para êxito no seu funcionamento.

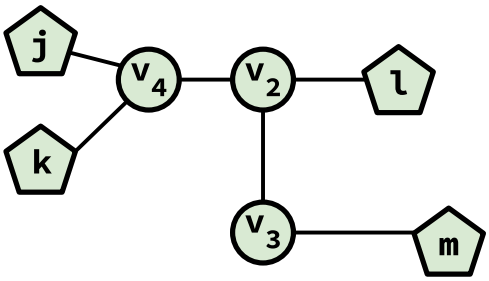
Figura 7. Redes lógicas da Topologia de Referência.



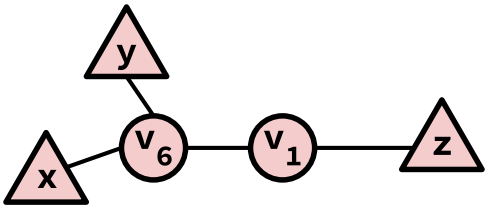
(a) Domínio de *broadcast*.



(b) Árvore de conexão da sub-rede 1.



(c) Árvore de conexão da sub-rede 2.



(d) Árvore de conexão da sub-rede 3.

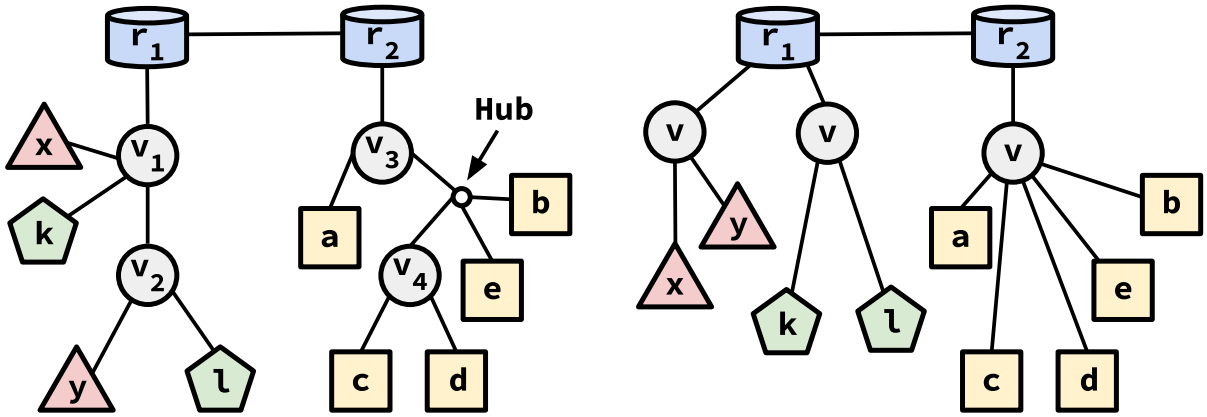
Fonte: Adaptado de (BEJERANO, 2009).

A resolução nível-interface considera a interface de conexão lógica associada à camada 3, podendo ter associados diversos endereços lógicos para uma mesma interface física. O uso do *traceroute* como método ativo utiliza pacotes com um TTL (*Time-To-Live*) limitado. O TTL é um campo do pacote que determina quantos dispositivos intermediários podem ser atravessados antes do destino ser considerado inalcançável. Os pacotes do *traceroute* são enviados de uma fonte para um destino, resultando na descoberta sucessiva dos endereços IP de interfaces de roteadores ao longo do caminho. O *traceroute* é uma ferramenta de diagnóstico de alcance genérico. Seu uso para descobrir a topologia de nível-interface é uma consideração que surgiu posteriormente, estando sujeito a falhas (MARCHETTA et al., 2013). Um roteador pode responder à uma mensagem do *traceroute* de formas diferentes dependendo do fabricante. Por exemplo, as respostas diferentes no caso de recebimento de mensagens com TTL igual a zero destacadas em (MOTAMEDI; REJAIE; WILLINGER, 2015). A resposta do roteador ao *traceroute* também pode ser diferente dependendo da configuração realizada pelos operadores da rede. Essas diferenças dificultam a criação de uma técnica de descoberta de topologia de rede baseada em mensagens sem padrão definido.

A descoberta de topologia em uma rede local pode apresentar resultados diferentes das ligações físicas entre dispositivos contidos na rede analisada. A Figura 8a ilustra uma topologia que identifica os dispositivos abaixo da camada de rede (L3). Nessa figura é possível identificar os *switches* v_1 ao v_4, *hub* e *hosts* em redes diferentes. Os dispositivos representados pelos *hosts* k e l são considerados estarem no mesmo domínio de difusão (L2) dos dispositivos x e y, porém em redes IPs diferentes (L3). Os dispositivos a ao e são considerados também como pertencentes a mesma rede IP e domínio de difusão. Na Figura 8b a mesma topologia pode ser observada em detalhamento nível-interface. Nessa figura os dispositivos sem endereço IP não são precisamente identificados, levando a descoberta incorreta da topologia. Cada interface IP do roteador r_1 é identificada como uma interface distinta e os switches não são corretamente identificados. Nesse cenário são considerados *switches* não gerenciáveis, incapazes de fornecerem informações de suas tabelas AFTs, para os possíveis erros na descoberta de topologia serem apresentados.

Apesar do *traceroute* ser utilizado como método ativo em diversas técnicas, para descoberta de topologia no nível-interface, não é explorado neste trabalho devido a formação da rede ser composta por *switches* e *hubs*. Esses dispositivos possuem a característica de serem transparentes à ferramenta *traceroute* e ping, protocolos L3, por serem dispositivos característicos L2. Neste trabalho a descoberta é feita com base no detalhamento nível-interface, com o emprego do protocolo ICMP e ARP para atualização dos dados. Após a identificação dos nós com IP, o algoritmo *Skeleton-tree* entra em ação para identificar os dispositivos de camada 2 mencionados por meio da análise dos dados coletados, sendo o *traceroute* desnecessário. A técnica de descoberta do algoritmo é apresentada na Seção 3.1. O tempo de descoberta das topologias é analisado antes e após as modificações

Figura 8. Falha na representação de topologia nível-interface



(a) Físico (b) Nível-interface

realizadas no processo de coleta de dados.

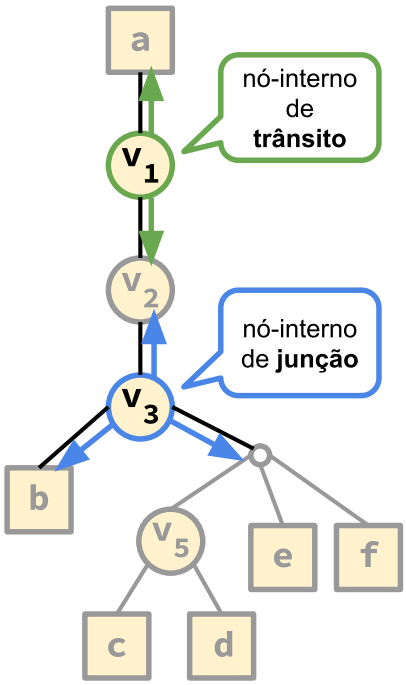
3.5 A árvore *Skeleton-tree*

As técnicas pesquisadas, e trabalhos relacionados, representam os elementos das topologias de rede em estruturas de árvores, um tipo especial de grafo. Várias das operações realizadas sobre estados intermediários das árvores até se chegar na versão final da topologia também têm como inspiração a Teoria de Grafos (SZWARCFITER, 2018). Frequentemente os nós das topologias de redes são tratados como vértices e os enlaces como arcos ou arestas. Como forma de distinguir entre a topologia de rede a ser analisada e a topologia sendo descoberta, as denominações nós e enlaces são utilizadas para fazer referência à rede real (ou simulada, como nos nossos testes), antes da descoberta. Nas referências à árvore *Skeleton-tree*, em (BEJERANO, 2009), são usados os termos vértices e arcos. A árvore *Skeleton-tree* é composta, assim, de vértices e arcos criados com base na árvore de sub-rede, na qual cada vértice corresponde a um nó ou um conjunto de nós da árvore de sub-rede. Mantivemos essa nomenclatura.

A árvore de sub-rede contém os nós-internos e nós-folhas envolvidos na sub-rede em análise. Os nós-internos da árvore de sub-rede são classificados em dois tipos ao serem incluídos na árvore *Skeleton-tree*, podendo ser de trânsito ou de junção. O nó é considerado como nó-interno de trânsito quando possui apenas 2 ligações distintas, ou seja, duas portas diferentes associadas à mesma sub-rede. Quando o nó-interno possui 3 ou mais ligações na sub-rede, ele é considerado como um nó de junção. Essa definição é representada visualmente na Figura 9, na qual o nó v_1 representa nó-interno de junção e o nó v_3 um nó-interno de trânsito.

Os nós-internos de trânsito que sejam consecutivos se unem em um único vértice,

Figura 9. Tipos de nós-internos.



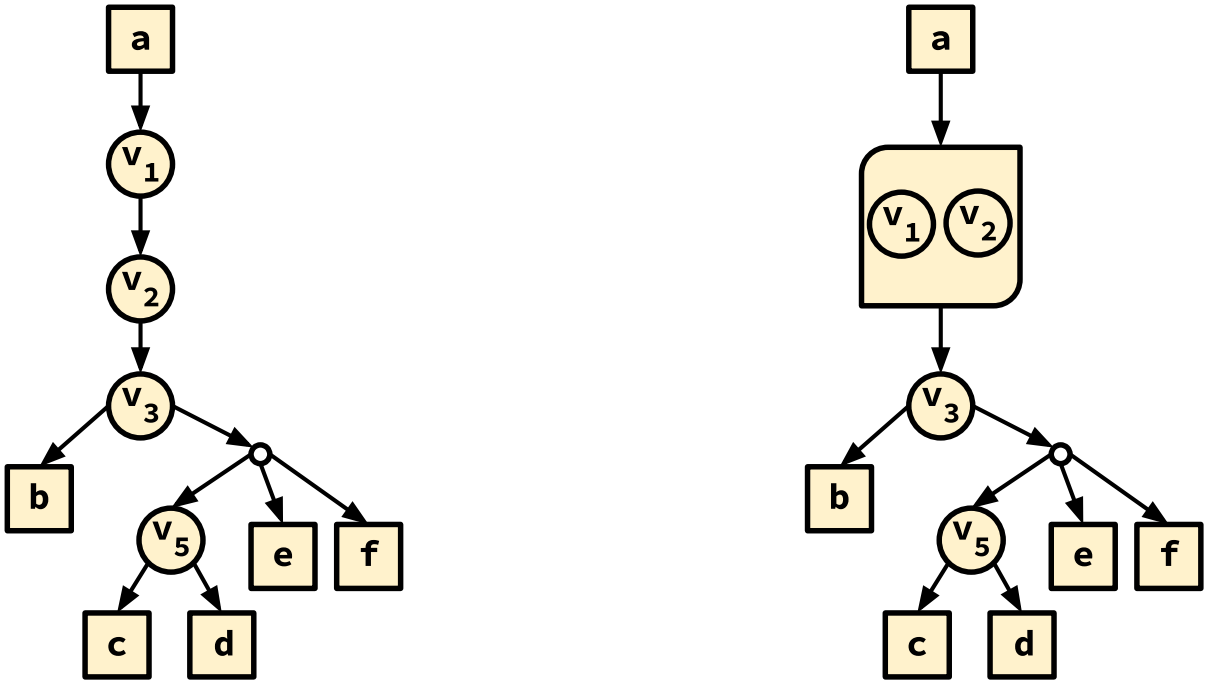
por terem o mesmo número de filhos, não sendo possível identificar inicialmente a sua ordem correta. enquanto que os nós-internos de junção e os nós-folhas são representados por vértices distintos. Desse modo, as árvores de sub-rede e sua árvore *Skeleton-tree* são apresentadas nas Figuras 10a e 10b respectivamente. Um nó ou vértice é denominado âncora quando existe a relação 1 para 1 entre o nó e o vértice. Os nós-internos de junção e nós-folhas em uma topologia analisada são sempre âncoras. Uma topologia é considerada única quando todos os seus nós são âncoras.

3.6 O Algoritmo

O algoritmo *Skeleton-tree* inicia a sua análise, conforme mencionado, por sub-rede, definidas por seus endereços lógicos (IPs). Para cada sub-rede é construída, então, uma estrutura de dados chamada árvore *Skeleton-tree* com base nos elementos que compõem a árvore da sub-rede que por sua vez é montada diretamente por operações do algoritmo sobre a spanning-tree e AFTs. Na primeira etapa da execução do algoritmo *Skeleton-tree* são construídas árvores *Skeleton-trees* para cada sub-rede conectada ao *host* de gerência. Na segunda etapa ocorre a unificação dessas árvores em uma única árvore *Skeleton-tree* que representa a topologia descoberta.

A notação usada na discussão das duas etapas do algoritmo é apresentada na Tabela 2.

Figura 10. Árvore de topologia da rede local



(a) Árvore de sub-rede 1

(b) Árvore *Skeleton-tree*

Fonte: Adaptado de (BEJERANO, 2009).

Tabela 2. Variáveis e notação adotada no algoritmo.

(v, k)	porta k do nó $v \in V(v_1, v_2, ...)$
$F_{v,k}$	entradas AFTs da porta (v, k) (nós alcançados)
D_v	portas ativas de v
$v(u)$	porta do nó v que leva ao nó u
N	nós-folhas exclusivos da árvore de sub-rede
V^N	nós-folhas e internos envolvidos na sub-rede N
$F^N_{v,k}$	entradas AFTs dos nós N na porta (v, k)
D^N_v	portas ativas de v para nós em N
r	nó raiz v
B_v	nós em N descendentes do nó v
$\|B_v\|$	quantidade de nós em B_v
n_v	valor do nó $v \in V^N$
$H(Y, A)$	árvore *Skeleton-tree*
C_y	nós representados pelo vértice y
Z	conjunto de arcos-fronteira
X	nós âncoras na árvore

Fonte: Adaptado de (BEJERANO, 2009).

Primeira etapa - Geração de árvore *Skeleton-tree*. O Algoritmo 1 apresenta a primeira etapa, que executa o cálculo para criação da estrutura de árvore *Skeleton-tree* e a definição dos nós âncoras identificados. Nas linhas 1 a 8 do Algoritmo 1 é realizada o cálculo do valor (n_v) atribuído para cada nó da árvore de sub-rede, a ser incluída na árvore *Skeleton-tree*, excluído o nó raiz. O valor é calculado com base no seu tipo (junção ou trânsito) e quantidade de descendentes que possui em relação ao nó raiz. Essa etapa tem como dados de ponto de partida os nós-folhas da sub-rede (N), os nós-internos (*switches* gerenciáveis) (V^N) e os dados das AFTs. Os nós-folhas e nós-internos são conhecidos antecipadamente, resultado dos passos de atualização das AFTs e coleta de dados SNMP, querendo se descobrir os nós não gerenciáveis e a topologia da rede. O descobrimento dos nós-folhas é realizado com o envio de mensagens ICMP (*ping*) para todos os IPs da sub-rede, identificando por meio da resposta os nós ativos existentes. A varredura de todos os IPs se faz necessária por não se ter conhecimento antecipado dos nós ativos na rede ou para detectar nós intrusos. O descobrimento dos nós-internos ocorre de forma semelhante, porém com endereços IPs da sub-rede destinada aos *switches* gerenciáveis. Os nós-internos possuem a característica de estarem executando o protocolo SNMP, sendo essa a forma de diferenciação dos nós-folhas. Apesar do processo de descobrimento dos nós-internos ser semelhante aos nós-folhas os seus atributos são distintos, sendo feita a diferenciação para a estrutura criada na execução do algoritmo. Utilizando o protocolo SNMP é realizada a coleta dos dados das AFTs após o descobrimento dos nós-internos. O SNMP coleta por meio das MIBs os dados dos objetos que contem a AFT e a identificação das portas do dispositivo. Em seguida é criada uma lista (L) decrescente ordenada pelo valor atribuído aos nós (linha 9).

O nó raiz (*root node*) é então utilizado como origem para a formação da árvore não direcionada da sub-rede. Os nós da lista são removidos sequencialmente para criação de vértices da árvore *Skeleton-tree*. No passo seguinte (linhas 10 e 15) é feita a criação do primeiro vértice (y) da árvore *Skeleton-tree* contendo o nó raiz (r). Nesse passo também são criados os arcos-fronteira (a) para o nó raiz, associando os seus descendentes (B_a). Um arco-fronteira consiste no arco criado que inicialmente contém apenas uma ponta conhecida, enquanto o nó conectado a outra ponta ainda não é descoberto. Os arcos-fronteira são armazenados pelo algoritmo em um conjunto (Z) separado e removidos à medida que a outra ponta é descoberta. A criação do primeiro vértice pode ser observada na Figura 11a para a primeira sub-rede da topologia de referência (Figura 7).

A execução do algoritmo prossegue com a retirada de forma iterativa de nós da lista (L) criada, linhas 16 a 50, como exemplo do nó v_2 mostrado na Figura 11b, realizando a criação dos arcos-fronteira associados. Cada nó retirado da lista é comparado com o nó anterior (linha 20).

Caso o nó tenha o mesmo valor atribuído do nó anterior e é um nó de junção, ele é associado ao mesmo vértice (linha 21). Na Figura 11c pode ser observado a ocorrência

desse processo entre os nós v_2 e v_1, no qual possuem mesma quantidade de descendentes (b ao f) e são nós-internos de junção, sendo assim ambos representados por um único vértice. A representação por um único vértice foi a solução encontrada por Bejerano para identificar os nós de ordenação incerta no momento. Neste ponto o algoritmo não pode definir qual dos nós aparece ordenado primeiro com base nas informações apenas da sub-rede. Essa distinção é feita posteriormente analisando as informações agregadas das sub-redes relacionadas ao nó.

Caso o nó da lista não tenha o mesmo valor atribuído do nó anterior, é removido o arco-fronteira da lista de arcos (Z), criado um novo vértice contendo o nó analisado (linha 24) e também novos arcos-fronteira para suas portas descendentes (linhas 26 a 30. Se esse processo ocorrer e os descendentes (B_a) do nó forem iguais ao descendente ($B_{v'}$) do nó anterior, o arco-fronteira removido é ligado ao novo vértice (linha 31). Esse passo pode ser observado na Figura 11d para o nó seguinte da lista da topologia de referência, nó $v3$. O nó possui o mesmo conjunto de descentes dos nós v_2 e v_1, porém é um nó-interno de trânsito, possuindo consequentemente um valor (n_v) diferente e nós descendentes as-

Algoritmo 1. Formação da árvore *Skeleton-tree* (1ª etapa).

ENTRADAS
N: Nós exclusivos da sub-rede
V^N: Todos os nós (internos e folhas) da árvore envolvidos na sub-rede
r: nó raiz
$AFTs$: tabelas de encaminhamento dos nós-internos

SAÍDAS
$SkeletonTree$: Árvore *Skeleton-tree* calculada

ALGORITMO $SkeletonTree(N, V^N, r, AFTs)$
1. **para** (cada $v \in V^N - \{r\}$), **fazer** {*cada nó da árvore de sub-rede*}
2. | $B_v = \bigcup_{k \in D_v - \{v(r)\}} F^N_{v,k} \bigcup (\{v\} \bigcap N)$
3. | **se** ($v \in N$ ou $|D^N_v| \neq 2$), **então** {*define valor do nó*}
4. | | $n_v = |B_v| - 1/2$
5. | **senão**
6. | | $n_v = |B_v|$
7. | **fim se**
8. **fim para**
9. L =Lista de nós $V^N = \{r\}$ ordenados por n_v
10. Cria novo vértice y com $C_y = \{r\}$
11. $n_y = |N| + 1/2$
12. **para** (cada $k \in D^N_r$), **fazer** {*cria 1º arco-fronteira*}
13. | cria arco-fronteira a para y, $B_a = F^N_{r,k}$
14. | $Z = Z \bigcup \{a\}$
15. **fim para**

— *continua* —

Algoritmo 1. Formação da árvore *Skeleton-tree*. (continuação)

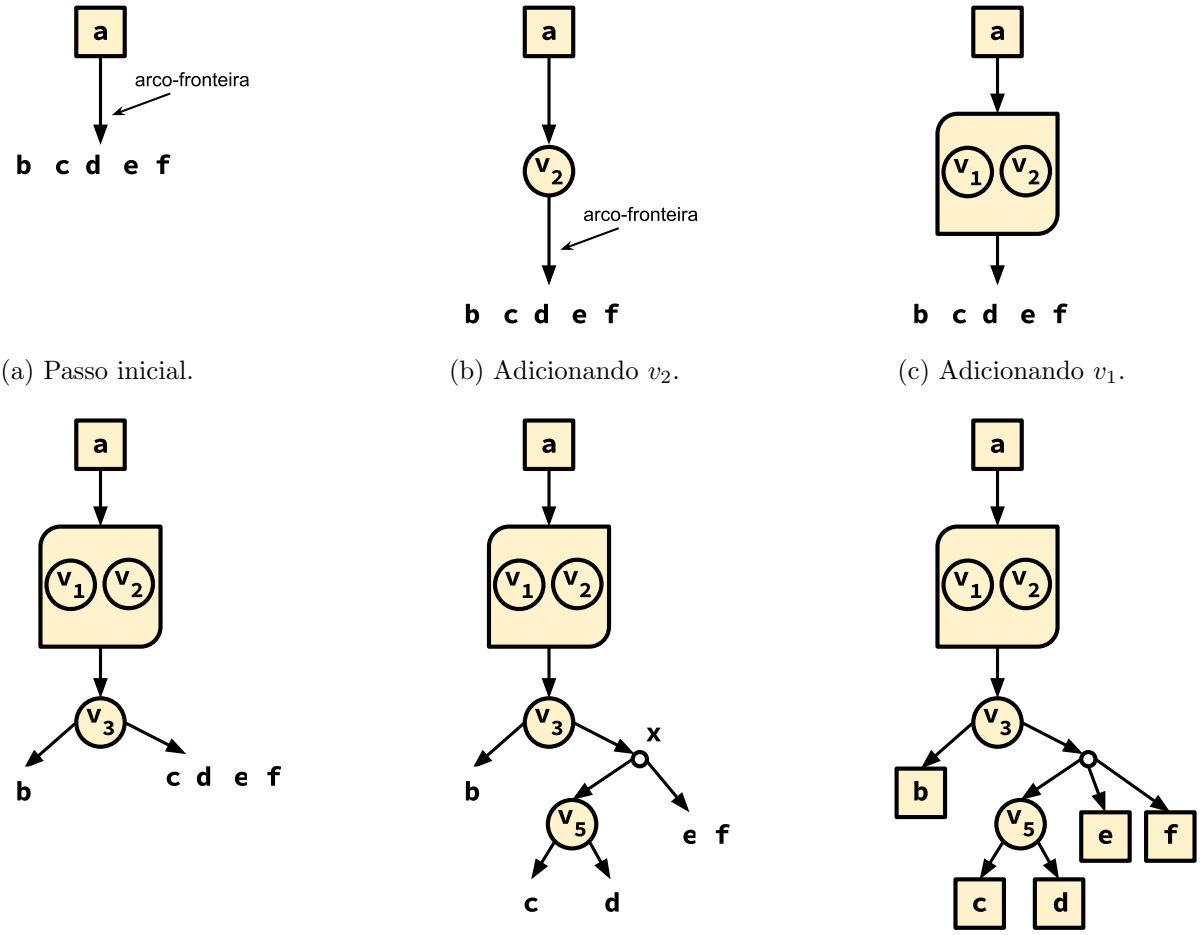
— *continuação* —

16. **enquanto** $(L \neq \emptyset)$, **fazer** {*Loop principal*}
17. | $v' =$ primeiro nó de L
18. | Encontra arco $a \in Z$
19. | y = ponto de partida de a em Y
20. | **se** $(n_y = n_{v'})$, **então**
21. | | $C_y = C_y \bigcup \{v'\}$
22. | **senão**
23. | | $Z = Z - \{a\}$
24. | | cria novo vértice y' com $C_{y'} = \{v'\}$
25. | | $n_{y'} = n_{v'}$
26. | | **para** (cada porta $k \in D^N_{v'} - \{v(r)\}$), **fazer** {*cria arcos-fronteira*}
27. | | | cria arco a' para y'
28. | | | $B_{a'} = B_{v'}$
29. | | | $Z = Z \bigcup a'$
30. | | **fim para**
31. | | **se** $(B_a = B_{v'})$, **então**
32. | | | conecta vértice y' ao arco a
33. | | **senão**
34. | | | **se** $(C_y = \emptyset)$, **então**
35. | | | | cria arco $\hat{a}$ para y
36. | | | | $B_{\hat{a}} = B_a - B_{v'}$
37. | | | | $Z = Z \bigcup \{\hat{a}\}$
38. | | | | conecta vértice y' ao arco a
39. | | | | $B_a = B_{v'}$
40. | | | **senão**
41. | | | | cria vértice x com $C_x = \emptyset$
42. | | | | $n_x = |B_a| - 1/2$
43. | | | | cria arcos a_1 e a_2 para x com $B_{a_1} = B_{v'}$ e $B_{a_2} = B_a - B_{v'}$
44. | | | | $Z = Z \bigcup \{a_2\}$
45. | | | | conecta vértice x ao arco a
46. | | | | conecta vértice y' ao arco a_1
47. | | | **fim se**
48. | | **fim se**
49. | **fim se**
50. **fim enquanto**

sociados a portas diferente, resultando na criação de um novo vértice conforme descrito.

Caso o nó da lista não tenha o mesmo valor atribuído do nó anterior, semelhante ao caso anterior, porém os descendentes são diferentes, o algoritmo identifica essa diferença (linhas 33 a 48) e cria um novo vértice (x) com as conexões associadas. Quando o nó em análise possui quantidade de descentes inferior ao nó anterior, o seu valor atribuído é menor. Isso indica que entre esse nó e o nó anterior existem nós-folhas conectados. Mesmo que não tenha sido identificado na varredura inicial, é necessário a existência de

Figura 11. Formação de árvore *Skeleton-tree* (1ª etapa).



(a) Passo inicial. (b) Adicionando v_2. (c) Adicionando v_1.

(d) Adicionando v_3. (e) Adicionando v_5 e x. (f) Adicionado nós-folhas.

Fonte: Adaptado de (BEJERANO, 2009).

um dispositivo para compartilhar o enlace e inserir tais os nós descendentes da diferença na topologia de rede. Esse dispositivo, até então desconhecido, é representado pelo novo vértice criado. Por sua vez, o vértice representa o dispositivo encontrado, *switch* não gerenciável ou *hub*. Processo equivalente pode ser observado na topologia de referência para nó v_5, apresentado na Figura 11e.

O passo final da criação da árvore *Skeleton-tree* ocorre com a adição dos nós-folhas aos seus arcos associados, conforme apresentado na Figura 11f.

Cada árvore *Skeleton-tree* passa pelo processo de estender as AFTs para que a unificação seja possível, combinando os dados dos nós âncoras coletados das demais *Skeleton-trees*, conforme apresentado no Algoritmo 2. Esse processo tem como entradas um vértice (y) representando o nó raiz, o conjunto de âncoras (X) da *Skeleton-tree* ($H(Y, A)$), que também é fornecida como entrada. Nas linhas 1 a 3 do Algoritmo 2, o processo de estender as AFTs é executado de forma recorrente para todos os vértices (y_j) descendentes

Algoritmo 2. Estender as AFTs.

ENTRADAS
y: vértice da árvore *Skeleton-tree*
X: nós âncoras da árvore *Skeleton-tree*
$H(Y, A)$: a árvore *Skeleton-tree*

ALGORITMO EstendeAFTs
1. **para** (cada descendente $y_j, j \in [1..J]$ de y), **fazer** {*descendentes de y*}
2. | $X_{y_j} = EstendeAFTs(y_j, X, H(Y, A))$
3. **fim para**
4. $X_y = (\bigcup_{j=1}^{J} X_{y_j}) \bigcup (X \bigcap C_y)$
5. **para** (cada $v \in C_y$), **fazer** {*atualiza AFTs*}
6. | $F_{v,v(r)} = F_{v,v(r)} \bigcup (X - X_y)$
7. | **para** (cada porta $v, k_j, j \in [1..J]$), **fazer**
8. | | $F_{v,k_j} = F_{v,k_j} \bigcup X_{y_j}$
9. | **fim para**
10. **fim para**

do vértice (y) fornecido. Assim, os dados são atualizados para todos os nós envolvidos (linhas 5 a 10). Esse processo ocorre tanto na primeira quanto na segunda etapa do algoritmo, toda vez após ser formada uma nova árvore *Skeleton-tree*, conforme pode ser observado no Algoritmo 3. O processo de estender as AFT possibilita a descoberta da topologia mesmo com a AFT incompleta. Isso é possível desde que a AFT coletada contenha o endereço do nó-raiz na porta que leva ao raiz (*root-port*) e as portas que levam aos nós-folhas estejam completas. Os demais nós-internos e nós-folhas são incluídos neste processo de estender a AFT. Quando o processo de estender a AFT é realizado na criação da *Skeleton-tree* (1ª etapa), os nós-folhas e nós-internos conhecidos por um nó-interno são incluídos nas AFTs dos demais nós-internos. Quando o processo de estender a AFT é realizado na unificação de 2 *Skeleton-trees* (2ª etapa), é identificada a ordem correta de nós de trânsito que tenham sido incluídas em um vértice. Por exemplo, a árvore *Skeleton-tree* apresentada na Figura 10b, resultante da sub-rede 1 (Figura 7b), possui um vértice representado pelos nós-internos de transito v_1 e v_2. Esses nós são considerados de trânsito para a sub-rede dos elementos a ao f. Entretanto, comparando com a árvore da sub-rede 2 (Figura 7c), o nó v_2 é um nó-interno de junção. No processo de unificação das *Skeleton-trees*, a extensão da AFT faz com que a AFT inicial da sub-rede seja combinada com outra sub-rede, identificando a ordem correta dos dispositivos.

Segunda etapa - Unificação das árvores *Skeleton-trees*. Essa etapa do algoritmo é iniciada após a criação das árvores *Skeleton-trees*, conforme procedimento descrito, para cada uma das sub-redes. O Algoritmo 3 apresenta o processo completo de descoberta de topologia, incluindo as chamadas aos algoritmos de cada etapa e ao procedimento para estender as AFTs (linhas 5 e 14). A unificação das árvores geradas é feita duas a duas

(linhas 8 a 15), criando uma nova árvore *Skeleton-tree* (linha 11), sendo executado o procedimento com base na consolidação dos nós âncoras comuns.

Algoritmo 3. Esquema de Descoberta de Topologia.

ENTRADAS
N^*: vértice da árvore *Skeleton-tree*
X: nós âncoras da árvore *Skeleton-tree*
$H(Y, A)$: a árvore *Skeleton-tree*

SAÍDAS
$SkeletonTree$(s): Árvore(s) *Skeleton-tree*(s) calculada(s)

ALGORITMO INFERIRTOPOLOGIA(V, N^*)
1. **para** (cada sub-rede $N_i \in N^*$), **fazer** $\{1^{\underline{a}}$ *etapa*$\}$
2. | $H_i(Y_i, A_i) = SkeletonTree(N_i, V^{N_i}, r_i, AFTs)$
3. | $X_i =$ âncoras de H_i
4. | seja y_i o vértice de H_i que representa r_i
5. | $EstendeAFTs(y_j, X_i, H_i(Y_i, A_i))$
6. **fim para**
7. **enquanto** (houver 2 skeleton-tree H_i e H_j com $X_i \bigcap X_j \neq \emptyset$), **fazer** $\{2^{\underline{a}}$ *etapa*$\}$
8. | $N_k = N_i \bigcup N_j$
9. | $r_k =$ qualquer nó em $X_i \bigcap X_j$
10. | $V^{N_k} = V^{N_i} \bigcup V^{N_j}$
11. | $H_k(Y_k, A_k) = SkeletonTree(N_k, V_{N_k}, r_k, AFTs)$
12. | $F_{v,v(r)} = F_{v,v(r)} \bigcup (X - X_y)$
13. | seja y_k o vértice de H_k que representa r_k
14. | $EstendeAFTs(y_k, X_k, H_k(Y_k, A_k))$
15. | Remove H_i e H_j
16. **fim enquanto**

A topologia de rede é definida sem ambiguidade quando todos os nós forem do tipo âncoras. Caso a unificação das *Skeletons-tree* não seja mais possível, o algoritmo encerra indicando falha na descoberta de topologia da rede.

3.7 *Skeleton-tree* com ARP

O uso do ICMP como método (ativo) para a atualização das informações a serem coletadas, conforme descrito nas Seções 3.2 e 3.3, utiliza mais dados no cabeçalho que o uso exclusivo do ARP. O cabeçalho extra do ICMP demanda mais etapas para ser processado em comparação ao ARP e consequentemente mais tempo para identificar endereços L2. Mesmo em uma rede formada exclusivamente por dispositivos L2, as mensagens ICMP da origem precisariam esperar a resolução do ARP caso o endereço MAC do destino tenha espirado em algum switch intermediário.

Dessa forma, é proposto o uso de mensagens de requisição ARP de forma exclusiva para atualizar os dados. O ARP trabalha em L2, sendo uma opção direta, sem gerar e processar o L3, para identificar e atualizar os endereços MAC (também L2) das AFTs. O uso do ARP também apresenta uma sobrecarga menor de protocolo quando comparado ao ICMP por não fazer uso dos cabeçalhos necessários do IP e do ICMP. O pacote ARP gerado não faz uso do IP específico de *host* de destino, utilizando o endereço de sub-rede como alvo. O pacote é gerado somente em L2 e o tratamento em L3 é feito apenas às respostas recebidas em L2, obtendo no momento da resposta a informação necessária do IP e MAC do dispositivo ativo. Assim se torna desnecessária a etapa de envio e resposta do ICMP, resultando na economia esperada de tempo e processamento.

A introdução do ARP como alternativa ao ICMP ocorre no processo de identificação de dispositivos ativos com endereçamento IP e da atualização inicial das AFTs durante a coleta de dados. Antes de iniciar a coleta de dados por SNMP, o método ativo com ARP atualiza novamente as AFTs dos nós-internos, tendo como ponto de origem o NMS, para cada nó previamente identificado na varredura inicial de IPs da sub-rede na topologia. Essa atualização também é realizada exclusivamente com ARP, garantindo dessa forma que as entradas AFTs que tenham sido sobrepostas ou descartadas sejam novamente atualizadas, evitando falhas no processo de descoberta.

3.8 **Implementação**

A implementação do código do algoritmo *Skeleton-tree*, descrito na Seção 3.1, foi desenvolvida em *Python*. Na implementação da técnica foram acrescentadas as alterações no método ativo, necessárias para se utilizar o ARP como alternativa ao ICMP, para atualização das AFTs e coleta de dados.

Quando definido originalmente o ICMP como método ativo para atualização de dados usados pelo algoritmo, a atualização da AFT é feita neste trabalho com o uso da ferramenta *nmap*, conforme linha 4 do trecho de Código 3.1 da função criada para a finalidade. O *nmap* é uma aplicação que tem a capacidade de gerar pacotes ICMP de forma concorrente para múltiplos destinos, permitindo a varredura simultânea de segmentos de endereços IPs.

No caso da utilização exclusiva do ARP como alternativa ao método ativo, o pacote a ser enviado na rede para atualização dos dados das AFTs é personalizado com o uso da biblioteca chamada *Scapy*. *Scapy* é uma biblioteca *Python* que permite a manipulação interativa de pacotes. Com a utilização dessa biblioteca é possível gerar e decodificar pacotes de um grande número de protocolos, sendo projetado para permitir criação rápida de pacotes usando valores padrão ou modificados. Dessa forma, é personalizado o pacote para o método ativo com o ARP, sendo criado de forma minimalista, contendo apenas os

cabeçalhos *Ethernet* e ARP (linha 14, Código 3.1).

```
1 /.../
2 if mode == 'ping':
3     alives = list()
4     alives.extend(ping_nmap(ips_rede, timeout))
5     if alives:
6         for ip in alives:
7             mac = getmac(ip)
8             mac_list.append(EUI(mac))
9             ip_list.append(IPv4Interface(ip))
10    /.../
11    arp_table_list = sorted(list(zip(ip_list, mac_list)))
12 elif mode == 'arp':
13    for end.rede in redes:
14        resposta = srp(Ether(end.difusao) / ARP(end.rede), timeout)
15        ips.append(IPv4Interface(resposta))
16        mac_list.append(EUI(resposta))
17    arp_table_list = sorted(list(zip(ips, mac_list)))
18 /.../
```

Código 3.1. Atualização da tabela ARP, escrito em Python.

Como dados de entrada são definidos os blocos de endereçamento IP (N), que representam as sub-redes, a serem analisados, identificando qual bloco IP é destinado aos *switches* gerenciáveis. Então os nós são descobertos, as AFT atualizadas e os dados coletados. O processo de identificação dos nós ativos e atualização das AFTs é realizado com o método específico escolhido, ICMP ou ARP, em tempo de execução. Os dados são coletados com SNMP e o algoritmo *Skeleton-tree* é executado para realizar a descoberta. No fim do processo de descoberta é feita a medição do tempo de descobrimento, para comparação entre a utilização de cada método para atualizar os dados, bem como a probabilidade de falha da descoberta.

Para cada interface do NMS associada a uma sub-rede diferente, um processo de atualização da sua tabela ARP é iniciado. São tentados todos os IPs da faixa da sub-rede para identificar os nós que estão ativos, por meio das respostas recebidas. Nesse ponto do código é possível escolher o método de descoberta ativa (ICMP ou ARP), a quantidade de pacotes enviados e o tempo limite de espera para considerar a requisição sem sucesso. A função *sk.arptable* (linha 5, utilizando *icmp*) é criada para enviar, receber e formatar as respostas do método ativo escolhido (*icmp* ou *arp*), associando o IP ao MAC de cada dispositivo da sub-rede, conforme trecho no Código 3.2.

Em seguida, a coleta SNMP é iniciada. Ao longo do processo de coleta é criada uma estrutura de dados com as informações solicitadas. No final desse processo todas as in-

formações coletadas estão contidas na estrutura criada. Essa estrutura é representada pela variável $SNMP\_DATA$ (linha 10) e fornece as informações de entrada $(N, V^N, r, AFTs)$ para o algoritmo *Skeleton-tree* (Algoritmo 1). Dessa forma é realizada a integração do passo de coleta com o passo da execução do algoritmo.

A verificação do sucesso no processo de descoberta é realizada com a comparação da topologia descoberta, resultante do algoritmo *Skeleton-tree*, com topologia utilizada para configurar o simulador, conforme apresentado nas Seções 4.2 e 4.3.

```
1 import skeleton as sk

2 sk.config_nms(redes)
3 /.../
4 for rede in redes:
5     arptable = sk.arptable(rede, probes, timeout, mode='icmp') \label{arptable-funcao-coleta}
6     ARP_TABLE_DATA[rede] = arptable
7     /.../
8     rede.set_all_nodes()
9 /.../
10 SNMP_DATA = dict()
11 for internal_node in internal_nodes:
12     atualiza(hosts)
13     SNMP_DATA.update(sk.get_snmp_data(internal_node))
14     /.../
15     inode.set_associated_subnets()
16 /.../
```

Código 3.2. Coleta de dados da tabela ARP, escrito em Python.

4 AVALIAÇÃO

Neste capítulo é apresentado uma avaliação da técnica implementada, utilizando o algoritmo *Skeleton-tree* na sua versão original e com a modificação proposta utilizando o ARP. Observa-se que os testes e as métricas avaliadas são mais amplas do que aquelas discutidas nas referências utilizadas (BEJERANO, 2006) e (BEJERANO, 2009).

Nas próximas subseções é apresentado o ambiente de teste e discutido como o GNS3 é empregado para replicar a topologia utilizada no trabalho original de Bejerano, bem como para gerar topologias aleatórias como entrada para os testes. Também são destacadas a integração do GNS3 e a nossa implementação em *Python*.

4.1 Ambiente de teste

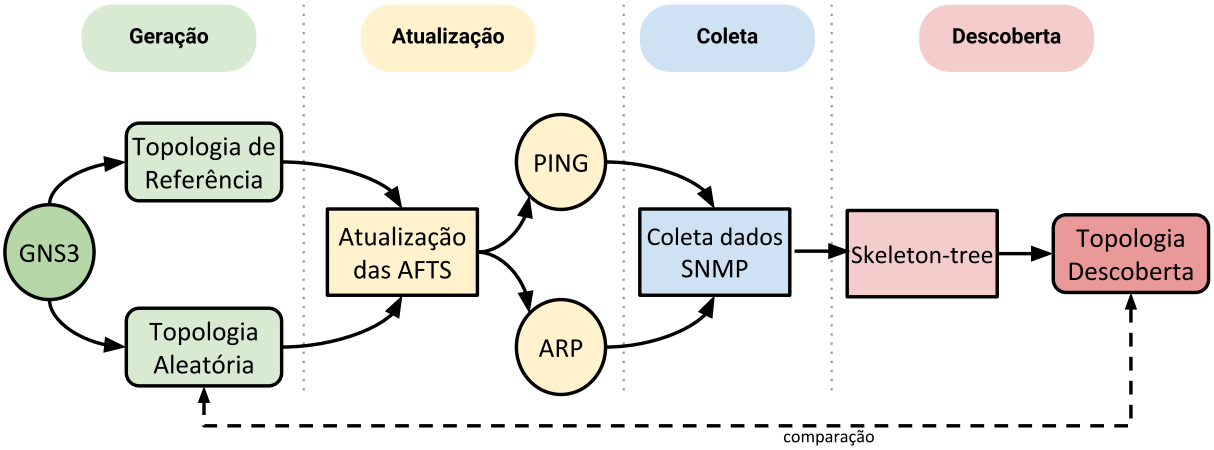
O ambiente de teste, representado na Figura 12, é composto por módulos para a geração de topologias, atualização das AFTs, coleta SNMP e descoberta. No primeiro módulo é produzido um ambiente virtualizado no GNS3 para simular uma rede real com base em uma topologia de referência. No segundo módulo, é feita a atualização das AFTs com utilização do *ping* ou ARP. A coleta é executada no terceiro módulo e no quarto módulo o algoritmo *Skeleton-tree* implementado em *Python* é executado para descobrir a topologia de referência, e topologias aleatórias, para validar seu funcionamento.

A geração das topologias para o teste e a implementação do *Skeleton-tree* são executadas em um computador com processador Intel Core i7-4771 e equipado 16GB de memória RAM. Nele são organizados e executados os elementos da simulação. Detalhes do hardware são apresentados na Tabela 3.

Tabela 3. Configuração de *hardware* usada nas simulações.

Processador	Intel i7 4471
Clock	3.90 GHz
Núcleos	4
Threads	8
Cache L2	1 MB
Cache L3	8 MB

Figura 12. Etapas do ambiente de teste proposto com *Skeleton-tree*



4.2 Topologias Avaliadas

Foram utilizadas diversas topologias para avaliar a técnica de descoberta com o *Skeleton-tree* e para medir o tempo para a topologia ser concluída. Inicialmente é manualmente configurada no simulador GNS3 a topologia de referência (Figura 13, Seção 4.2) para validar a técnica de descoberta. Em seguida são geradas 500 redes aleatórias, contendo 120 nós, com código auxiliar criado em *Python*, o uso do módulo NetworkX (HAGBERG; SCHULT; SWART, 2008) e as APIs do GNS3. As topologias aleatórias são utilizadas como base para configurar o simulador, calcular os resultados da descoberta e do tempo decorrido com o uso do ARP e ICMP.

Após configuração do GNS3 conforme recomendações dos desenvolvedores, os dispositivos são introduzidos por meio dos *appliances*. Para orquestrar as etapas descritas na Figura 12 é utilizado o NMS, que agrega as funções de criar a rede no GNS3, iniciar o método ativo (ICMP e ARP) de atualização dos dados das AFTs, iniciar o método passivo (SNMP) para coleta de dados, executar o algoritmo *Skeleton-tree* e medir tempo total da técnica para descoberta da topologia. O processo de criar a rede no GNS3 foi automatizado em uma rotina adicional elaborada em *Python* para gerar redes aleatórias e as configurar no simulador. A interface do código *Python* com o GNS3 é desenvolvido com as recomendações e APIs publicadas pelos desenvolvedores.

Topologia de Referência. A topologia de referência é replicada com o uso do GNS3 e seus *appliances*. É escolhida a opção de criação de *appliances* próprios, no qual são implementas as funções básicas de um dispositivo de rede operacional. Os *appliances* nativos para *hub*, *switch* não gerenciável e *hosts* (VPCs) também são utilizados. Um *appliance* é criado para representação dos *switches* gerenciáveis e outro *appliance* para representar o NMS, acrescentando-os a rede. O *appliance* para o NMS é gerado com base em uma

Tabela 4. Configuração de *software* usada nas simulações.

SO hardware	Windows 10 64x versão 1713.407
GNS3 Local	versão 2.1.11
GNS3 VM	versão 2.0.3
NMS VM	Linux Lubuntu 17.10
Python	versão 3.6.5
VMware	12.5.9 build 7535481

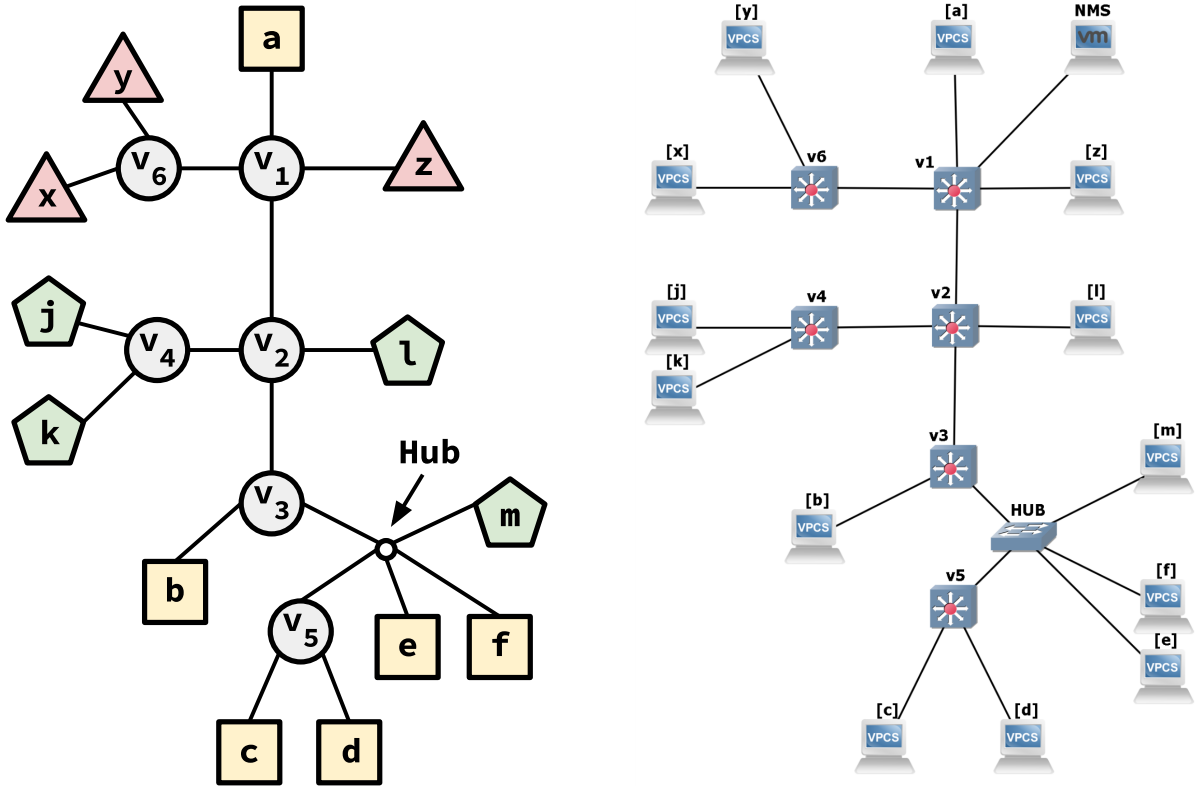
máquina virtual Linux que é adotada como estação de desenvolvimento dos códigos em *Python*. O NMS agrupa as funções de geração das topologias aleatórias, configuração das topologias no GNS3, geração de dados para atualização das AFTs, coletas SNMP e cálculos do algoritmo *Skeleton-tree*. Essa máquina virtual é configurada com Linux Lubuntu 17.10 com 6GB de memória RAM e o ambiente *Python* necessário para desenvolver o código. A configuração de software é apresentada na Tabela 4.

A VM GNS3 local é utilizada para criar e executar as simulações, seguindo recomendação dos desenvolvedores do software. Essa versão do GNS3 consiste em uma máquina virtual previamente configurada e otimizada com GNS3, sendo integrada facilmente com processos GNS3 em execução em outras máquinas.

A virtualização do sistema operacional do fabricante Cisco, conhecido como Cisco IOS (*Internetwork Operating System*), é adotado como base para o *appliance* que representa os *switches* gerenciáveis. É escolhida uma versão do IOS que trabalha com *switches* L3. Esses *switches* são capazes de realizar funções de comutação e também roteamento. As funções de roteamento são desativadas, apesar de disponíveis, sendo mantida as funções de gerenciamento e comutação. Dessa forma, os *switches* emulados se comportam como *switches* gerenciáveis de camada 2. A escolha é feita levando em consideração o domínio técnico nos equipamentos da Cisco e a sua ampla utilização no marcado. Outro fator na escolha da versão com suporte à L3 é a possibilidade de dar continuidade ao trabalho aqui proposto sem necessidade de grandes alterações no código implementado. Por se tratar do mesmo modelo de dispositivo utilizado para comutação, o roteamento pode ser habilitado de acordo com a necessidade. Apesar de se tratar de um sistema exclusivo do fabricante, somente as funções comuns aos *switches* e MIBs genéricas são utilizadas. Protocolos e funções proprietárias são desabilitados.

Na criação da rede no GNS3 da topologia de referência gerada, primeiro os *switches* são importados em forma de *appliances* e configurados, seguidos dos *hosts VPCs* e *hubs*. Os dispositivos são interconectados após criados na topologia. A topologia de referência, contendo 4 sub-redes, é apresentada na Figura 13a. Os *switches* gerenciáveis são simbolizados pelos dispositivos v_1 à v_5. As demais sub-redes são representadas pelos dispositivos a ao f, k ao m e x ao z respectivamente. A topologia equivalente implementada no GNS3, com o NMS conectado, é apresentada na Figura 13b.

Figura 13. Topologia



(a) Referência

(b) GNS3

Fonte: Adaptado de (BEJERANO, 2009).

Topologias Aleatórias.

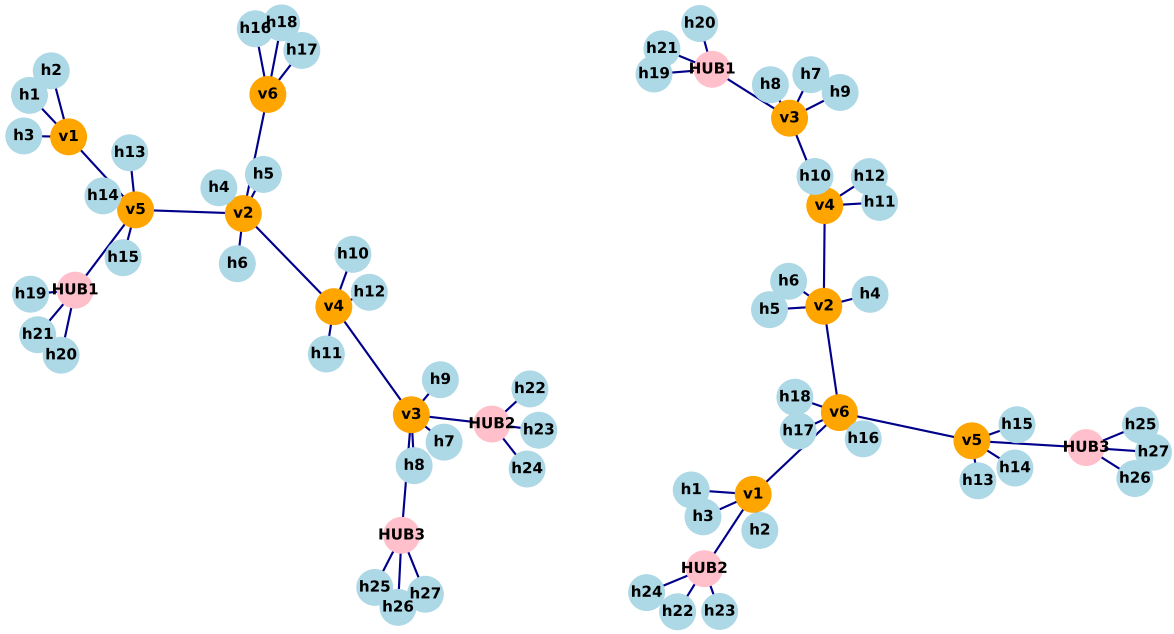
A criação automatizada das topologias aleatórias ocorre por meio de algoritmo próprio elaborado, independente da técnica *Skeleton-tree*. O algoritmo próprio é elaborado em *Python*, com suporte do pacote NetworkX, no qual são criadas redes aleatórias. O NetworkX é um pacote para a criação, manipulação e estudo da estrutura, da dinâmica e das funções de redes complexas.

A criação das redes aleatórias segue as instruções de Bejerano, na qual topologias análogas são geradas com nós diferenciados para *switches*, *hubs* e *hosts*, conforme topologia exemplo apresentado na Figura 14a. Inicialmente são geradas árvores aleatórias formadas por *switches* devidamente interconectados, representados na Figura 14a pelos nós v_1 ao v_6. Em seguida, são conectados aleatoriamente o número necessário de hubs (HUB_1 ao HUB_3) às portas livres dos switches e posteriormente os hosts (h_1 ao h_{27}), de forma uniforme aos switches e aos hubs da árvore construída. Por fim, cada host é arbitrariamente associado a uma única sub-rede, sendo à árvore construída conectada ao NMS por uma interface dedicada a cada sub-rede. Com o objetivo de estudar a eficácia em relação a quantidade de sub-redes, foram criados cenários com 3 a 7 sub-redes. A atribuição dos *hosts* da topologia a cada sub-rede ocorre pelo endereçamento gerado automaticamente pelo NMS para o simulador GNS3 e atribuído a cada *host*. Dessa forma, cada *host* de uma sub-rede pode se comunicar com o endereço do NMS dedicado ao seu bloco IP e demais dispositivos da sua sub-rede.

Cada rede aleatória considera instâncias de rede definidas por 10 *switches*, 10 *hubs* e 100 *hosts*, gerados. As informações das topologias geradas de forma aleatória são armazenadas e resgatadas posteriormente uma a uma. Cada rede aleatória é replicada em uma rede virtualizada no GNS3 para executar o algoritmo *Skeleton-tree*.

A quantidade de nós-internos, representada pelos *switches*, e nós-folhas, representadas pelos hosts, é mantida constante para cada rede aleatória. O código auxiliar criado é então utilizado para configurar cada ligação da topologia aleatória para o GNS3, na qual são definidas as sub-redes e o endereçamento adequado para cada nó-folha ou *host*. O código auxiliar realiza a divisão de endereços de forma automática e proporcional em relação à quantidade de sub-redes para analisar, distribuindo de maneira uniforme os endereços e configurando cada nó-folha. Os switches gerenciáveis, ou nós-internos, não necessitam serem reconfigurados a cada nova rede aleatória carregada por estarem contidos em sub-rede distinta, agindo de forma semelhante independente das suas conexões ou quantidades de sub-redes carregadas. O código auxiliar realiza a criação e configuração dos nós e enlaces no simulador por meio das APIs do GNS3 disponibilizada por seus desenvolvedores, utilizando estruturas de dados baseadas no formato JSON. A comunicação do código auxiliar com o GNS3 para a transferência de dados se faz por meio de métodos GET e POST do HTTP, responsáveis por coletar informações do simulador e fornecer os dados necessários à API.

Figura 14. Topologias geradas com NetworkX, plotadas com auxílio do Pyplot (Matplotlib) em código próprio.



(a) Árvore aleatória 1

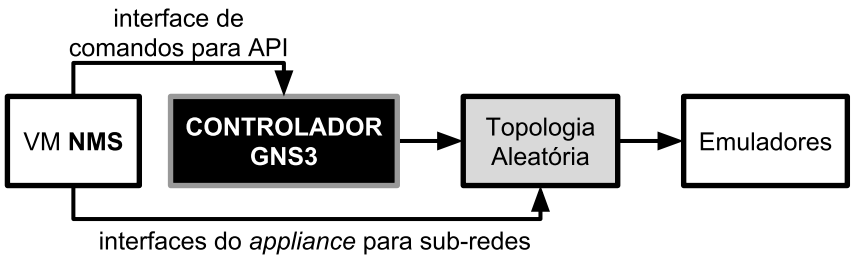
(b) Árvore aleatória 2

4.3 Integração do NetworkX, GNS3 e Python

As diversas topologias aleatórias mencionadas na Seção 4.2 que são utilizadas nas simulações de rede precisam geradas e configuradas no GNS3. Esse processo é feito de forma automatizada, por meio do NMS, devido a sua complexidade. Assim, é necessária a integração do método criado para gerar as topologias aleatórias com o método criado para configurar de forma autônoma tais topologias no simulador. Dessa forma, a integração das topologias aleatórias geradas no NetworkX com o GNS3 é feita também com o auxílio do *Python*.

O NMS é implementado como uma VM independente. Essa VM é incluída no GNS3 em forma de *appliance*, fazendo a comunicação direta com a topologia pelas interfaces locais criadas para cada sub-rede. A comunicação do NMS com o controlador GNS3 para a criação das topologias aleatórias geradas no NetworkX é feita por meio de uma interface local própria para esse fim. Essa interface é utilizada para enviar os comandos ao controlador GNS3 por meio de suas APIs, configurando os nós e enlaces das redes aleatórias geradas. O esquema de integração é apresentado na Figura 15, na qual se observa a ligação da interface local do NMS com o controlador GNS3 para reprodução de comandos da API e as interfaces ligadas por meio do *appliance* para cada sub-rede da topologia.

Figura 15. Esquema da integração do NetworkX com o GNS3.



Foi implementado uma rotina auxiliar que usa a estrutura de árvore gerada com suporte do software NetworkX (Seção 4.2) para fazer a configuração da rede no GNS3, conforme trechos apresentados no Código 4.1. Nas linhas 3 a 5 são gerados grafos de árvores aleatórias. A função *gera_arvore* (linha 4) forma, com auxílio do NetworkX, grafos das árvores aleatórias conforme definido na Seção 4.2 e exemplificado na Figura 14. Em seguida é configurado a interface para interagir com o GNS3 (linha 7), para envio de comandos de criação e remoção de recursos da topologia por meio de função criada para utilizar suas APIs. A adição e remoção dos recursos (linhas 9 a 14) é realizada primeiramente com a remoção de possíveis enlaces existentes (linha 10) de simulações anteriores, mantendo a conexão com o NMS. São criados então os nós (linha 11) com base em um dos grafos gerados e no número de sub-redes programado. O grafo também é utilizado como base para criar os enlaces (linha 12) no GNS3. O processo de criação da topologia é concluído com a inicialização de todos os nós que participam da simulação (linha 13).

```
1 /.../
2 grafos = list()
3 for i in range(500):
4     arvore_aleatoria = gera_arvore(n_switches, n_hubs, n_hosts)
5     grafos.append(arvore_aleatoria)
6     /.../
7 pc = gb.Gns3(ip, project_id)
8     /.../
9 for grafo in grafos:
10    pc.limpa_enlaces(keep=(nms_id,))
11    pc.nodes_do_grafo(grafo, n_subnets)
12    pc.enlaces_do_grafo(grafo)
13    pc.inicia_nodes()
14 /.../
```

Código 4.1. Integração NetworkX e GNS3, escrito em Python.

4.4 Testes Realizados

Para avaliar a implementação desenvolvida, foram realizados testes com a topologia de referência e em mais 500 topologias aleatórias, contendo 120 nós. Desta forma, foi avaliado o funcionamento da reprodução do algoritmo e posteriormente feita a análise dos resultados da técnica de descoberta nas diferentes redes aleatórias, para analisar os cenários com a modificações propostas no método ativo.

Tomou-se a topologia de referência como ponto de partida para a realização dos testes. Inicialmente são testados no GNS3 diversos *appliances* para escolher entre os que apresentam os recursos necessários. São verificados se o *hardware*, as interfaces e o funcionamento dos protocolos virtualizados estão em conformidade com os equipamentos reais. Validado o funcionamento, é escolhido o *appliance* que consome menos recursos de *hardware*, esperando assim deixar mais recursos disponíveis para o processo de descoberta. Essa escolha é feita de forma preventiva, porém a execução dos testes não fez se aproximar do esgotamento dos recursos do *hardware* disponível (Seção 4.1).

Após o processo de validação dos recursos físicos e lógicos, é criada a topologia de referência (Figura 13a, Seção 4.2) e executada a técnica de descoberta com o algoritmo *Skeleton-tree* para testar sua eficiência. Concluído os testes e verificado o funcionamento da técnica é iniciado o processo com as topologias aleatórias.

As 500 redes aleatórias geradas são configuradas 1 a 1 no simulador. Então, o processo de descoberta com a técnica *Skeleton-tree* é realizado com ICMP e ARP para coleta de dados em cada rede configurada no GNS3. Os resultados foram avaliados quanto à acurácia da topologia descoberta em relação à gerada e o tempo de descoberta medido.

5 RESULTADOS

As probabilidades de falha do esquema de descoberta ao inferir a topologia completa, definidas por Bejerano em (BEJERANO, 2006) e (BEJERANO, 2009), são analisadas nas simulações de rede implementadas. A probabilidade de falha indica a proporção de redes que não podem ser descobertas do total de simulações executadas. É avaliado o impacto do tempo limite de espera, ou *timeout*, do método ativo de atualização da AFT na probabilidade de falha do esquema. O tempo limite de espera é definido como tempo limite que uma solicitação, ARP ou ICMP, aguarda pela resposta para considerar o dispositivo inexistente ou inacessível. Considera-se como falha a instância da rede simulada em que o esquema de descoberta de topologia não pôde inferir uma estrutura única de *Skeleton-tree*. Dada a topologia, esse processo não é determinístico, tendo em vista a aleatoriedade dos processos dos dispositivos na rede simulada. Um dispositivo que leve maior tempo para responder uma solicitação pode ser considerado erroneamente como inexistente, levando à uma possível falha no processo de descoberta. A falha pode ocorrer quando a análise das AFTs pode não definir uma estrutura única ou quando as AFTs estiverem incompletas. A falha também ocorre quando o esquema de descoberta de topologia não encontrou um nó âncora comum, conforme definido na Seção 3.1.

Os resultados das simulações são apresentados nos gráficos deste capítulo. Cada gráfico considera instâncias de rede definidas por 10 *switches* cooperativos (gerenciáveis), 10 *hubs* e 100 *hosts*, gerados conforme metodologia da Seção 4.2. Para possibilitar comparação, as quantidades são definidas em equivalência com o apresentado por Bejerano. Gerou-se gráficos que representam as probabilidades de falha do esquema à medida que o número de sub-redes e os *switches* não cooperativos aumentam em diferentes cenários de tempo limite de espera. Destacou-se as diferenças entre o processo realizado com o uso do ARP e do ICMP. Cada ponto dos gráficos corresponde à probabilidade de falha de 500 instâncias de topologias. Cada topologia possui as mesmas características de número de *switches*, *hubs* e *hosts* ligados de forma aleatória, conforme se apresenta na Seção 4.2. As probabilidades de falhas representam as topologias que não são descobertas com a técnica, utilizando o algoritmo *Skeleton-tree*, nos tempos limites de espera determinados para os métodos ativos ARP e ICMP de atualização de dados.

Definiram-se três cenários com valores para tempo limite de espera distintos de um, dois e três segundos respectivamente. Só são consideradas respostas recebidas dentro desses tempos limites de espera para cada cenário. Atrasos na propagação, tempo decorrido no processamento no dispositivo de destino ou nós intermediários podem resultar em respostas recebidas posteriormente ao tempo limite de espera. Nesses casos a resposta é desconsiderada. O tempo limite de espera é calculado de forma individual para cada pacote enviado pela estação NMS. O tempo é iniciado no momento que o pacote sai do

NMS e se aguarda a resposta até o tempo limite de espera. Caso a resposta não chegue, o destino é considerado como inexistente.

Tratando-se de um ambiente simulado por meio da virtualização de *switches* e *hosts*, o comportamento desses dispositivos é semelhante aos dispositivos físicos equivalentes. Cada dispositivo de rede passa pelo processo de receber os dados, processar seu conteúdo, consultar as tabelas e encaminhar ao destino.

Quando um dado ICMP é enviado a um dispositivo com endereço físico (MAC) desconhecido, o processo de descobrimento pelo ARP precisa ser realizado. Considerando nesse cenário uma sub-rede formada por *switches*, o tempo necessário para que uma resposta à solicitação ICMP seja atendida aumenta, devido a necessidade de aguardar o processo do ARP. Os demais *switches* intermediários aprendem o endereço MAC somente após o destino responder a solicitação de descoberta ARP, associando-o a uma interface, permitindo então a resposta chegar à origem. O tempo total decorrido nesse processo pode superar o tempo limite de espera do protocolo que está aguardando a resposta, levando a falha no processo de descoberta de topologia. As redes analisadas são configuradas e não é iniciado trafego para simular comunicação entre os dispositivos. O tempo é maior na primeira comunicação com cada dispositivo, não impactando significativamente nas comunicações seguintes, desde que o seu endereço registrado na AFT não tenha sido removido. Um endereço pode ser removido por sobreposição de novos endereços, aprendidos mais recentemente, ou por ociosidade na comunicação com o dispositivo.

Conforme visto, os resultados apresentados incluem a variação no método ativo de atualização de dados para descoberta, com a inclusão das atualizações das tabelas AFT por meio do uso do protocolo ARP, e a variação no tempo limite de espera em diversos cenários de descoberta de topologias. Em cenário com o tempo limite de espera do método ativo de 1 segundo a descoberta com atualização dos dados das AFTs por meio do ARP apresenta probabilidade de falha muito menor do que com o ICMP. Nos resultados apresentados por Yigal Bejerano em (BEJERANO, 2006) e (BEJERANO, 2009) não são apresentados a influência do tempo limite de espera na probabilidade de falha, sendo aqui identificado uma variação em relação ao ICMP quando o tempo é reduzido. Em cenários com tempo limite de espera maior ou com o uso do ARP esse comportamento não ocorre.

O resultado do processo de descoberta da técnica *Skeleton-tree* com o uso do protocolo ICMP e tempo limite de espera de 3 segundos na atualização e coleta de dados pode ser observado na Figura 16. O resultado com o uso exclusivo do ARP para as mesmas redes poder ser observado na Figura 17.

A probabilidade de falha se mostra crescente para descoberta de topologias com número maior de sub-redes e com mais de 4 *switches* não cooperativos. Esse comportamento é esperado para o protocolo ICMP, conforme mostrado por Bejerano em (BEJERANO, 2009). Este trabalho complementa os resultados de Bejerano, incluindo o ARP como um novo método ativo e avaliando a probabilidade de falha em relação ao tempo

Figura 16. Resultados das simulações de redes com ICMP e 3s de tempo limite de espera.

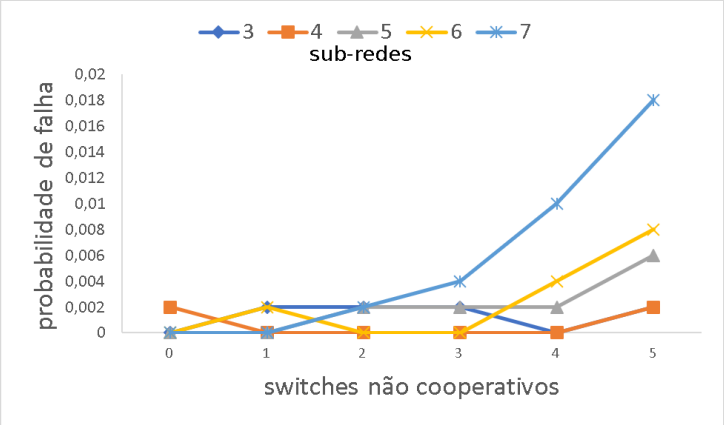


Figura 17. Resultados das simulações de redes com ARP e 3s de tempo limite de espera.

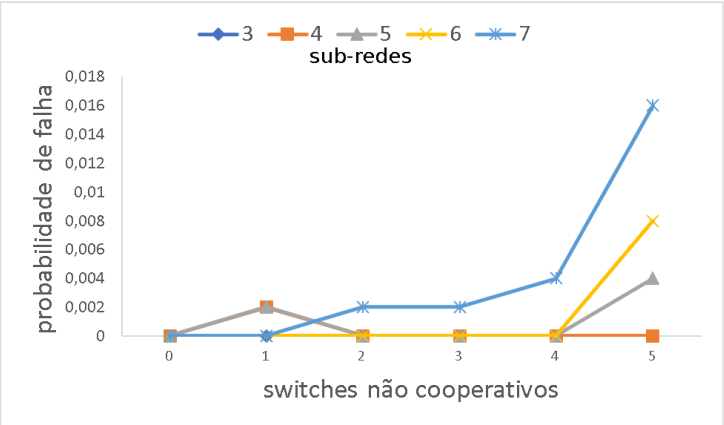


Figura 18. Resultados das simulações do ARP e ICMP para 3 sub-redes com 2s de tempo limite de espera.

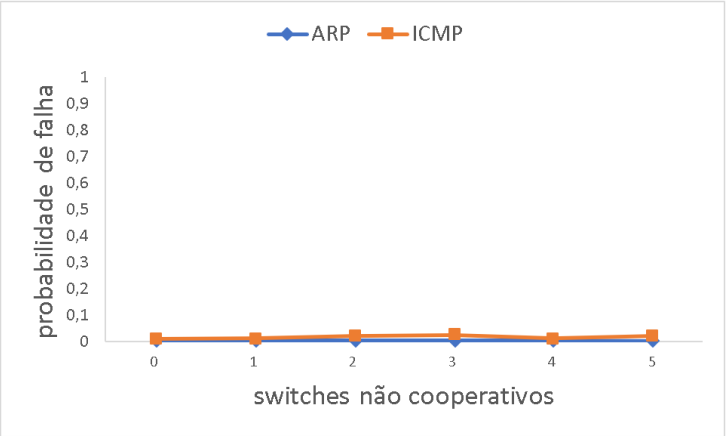
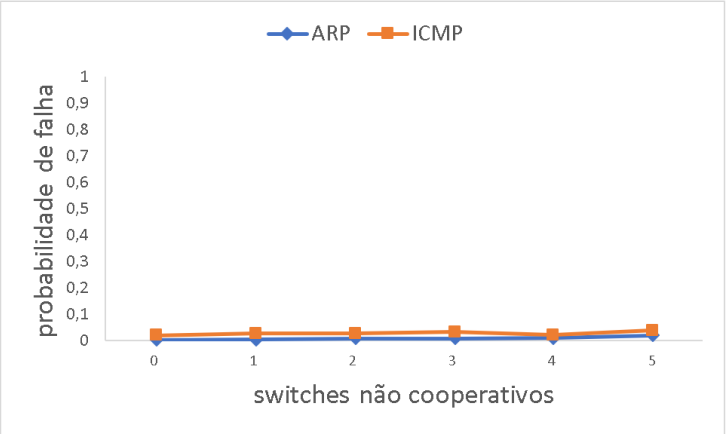


Figura 19. Resultados das simulações do ARP e ICMP para 7 sub-redes com 2s de tempo limite de espera.



limite de espera dos pacotes enviados para atualizar os dados. O mesmo comportamento do uso do ICMP se apresenta nesse cenário, com 3 segundo de tempo limite de espera, para o uso exclusivo do ARP como método ativo, indicando sucesso no preenchimento das tabelas AFTs com os dados necessários.

Os resultados das simulações utilizando tempo limite de espera de 2 segundos para o ARP e ICMP, apresentado na Figura 18, mostram probabilidades de falha semelhantes para ambos os protocolos quando utilizadas 3 sub-redes. Quando a quantidade de sub-redes aumenta para 7 nesses cenários, o resultado apresenta uma pequena vantagem para o ARP, conforme Figura 19.

A Figura 20 apresenta a probabilidade de falha no processo de descoberta de topologia para 3 sub-redes com tempo limite de espera de 1 segundo. As simulações nesse cenário apresentam grande variação na probabilidade de falha. Resultado semelhante

Figura 20. Resultados das simulações do ARP e ICMP para 3 sub-redes com 1s de tempo limite de espera.

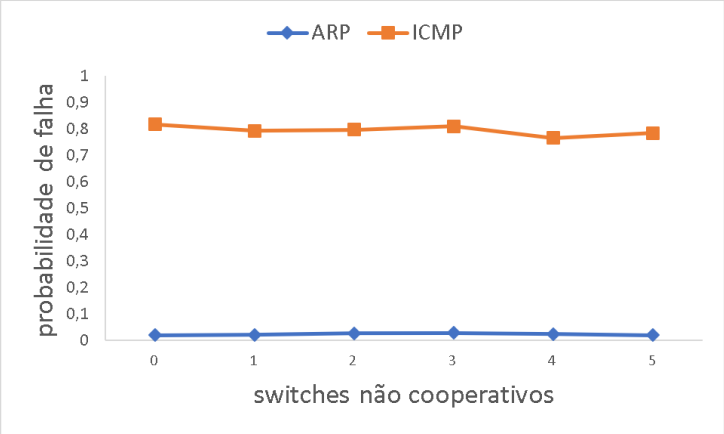
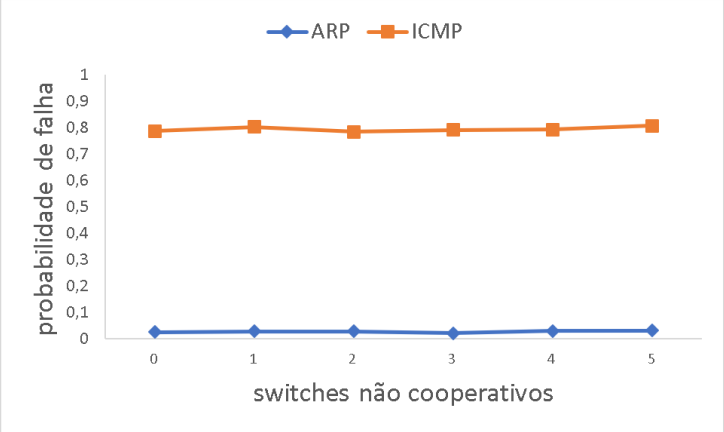


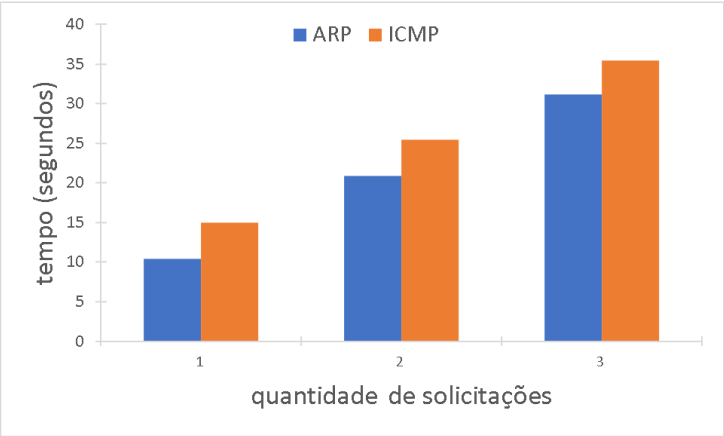
Figura 21. Resultados das simulações do ARP e ICMP para 7 sub-redes com 1s de tempo limite de espera.



ocorre quando a quantidade de sub-redes aumenta conforme pode ser visto na Figura 21. É possível notar que a quantidade de switches não cooperativos possui baixo impacto na probabilidade de falhas. Em ambos os cenários a utilização do ARP apresenta probabilidade de falha inferior a $0,1$ à medida que o ICMP apresenta valores em torno de $0,8$. Esse comportamento pode ser justificado levando em consideração o tempo de processamento dos dispositivos para responder as solicitações recebidas, incluindo a necessidade da suspensão do ICMP para utilizar o ARP como forma resolver endereços que não estão nas AFTs.

Uma forma encontrada de minimizar as diferenças observadas no aumento de probabilidade de falha é o incremento na quantidade de solicitações enviadas pelo ARP e ICMP. Nesse processo ocorre o incremento no tempo de descoberta, conforme apresentado na Figura 22, com resultados de tempo menores para o uso do ARP. Nessa figura,

Figura 22. Média do Tempo de Descoberta das topologias.



solicitações ARP e ICMP, com tempo limite de espera de 1s, são feitas aos nós-folhas antes de coletar as AFTs de cada nó-interno (switch gerenciável). Esse procedimento ocorre para forçar a atualização dos dados na AFT e garantir sua integridade no processo de varredura inicial. A solicitação, quando feita com ARP por difusão, alcança todos os nós-folhas simultaneamente, resultando em probabilidade de falha semelhante ao apresentado nas Figuras 20 e 21. A solicitação, quando feita com ICMP *unicast*, tem a quantidade de endereços da rede segmentados de acordo com a quantidade de solicitações. Todos os endereços de destino enviados de forma concorrente para uma solicitação, metade de cada vez em duas solicitações e a terça parte para três solicitações. Nos cenários com uma solicitação, a probabilidade de falha foi maior que o ARP, semelhante ao já apresentado nas Figuras 20 e 21. Nos cenários com duas e três solicita essa diferença não é observada, semelhante aos tempos limites de espera apresentados nas Figuras 18 e 19 .

Conforme analisado nos resultados apresentados, conclui-se que eventuais esgotamentos do tempo limite de espera nas respostas do processo de medição ativa interferem no resultado, influenciando em uma maior probabilidade de falha. Entretanto, o aumento do tempo limite de espera para o ICMP pode tornar os erros próximos. Observa-se também que a descoberta com ARP não demanda o tempo extra do ICMP por trabalhar na camada 2 do modelo OSI, mesma em que os *switches* trabalham. Isso faz com que seja eliminado a necessidade de passar para camadas superiores o processamento ICMP, que trabalha na camada 3. Por sua vez, o ICMP necessita acionar o protocolo ARP caso o destino não seja conhecido, ficando ocioso até que uma resposta seja recebida ou o tempo limite de espera atingido. Caso semelhante indica ocorrer nas Figuras 20 e 21 já mencionadas, tendo em consideração que a simulação é iniciada com a rede ociosa e sem a indução de tráfego de fundo entre dispositivos. Nesses cenários, a solicitação ICMP para cada IP de destino, com o nó existente ou não, necessita ser aguardar a resolução do endereço MAC feita pelo protocolo ARP.

Cabe mencionar que nos trabalhos citados não foram feitas relações com o tempo limite de espera e duração dos processos de descoberta, sendo introduzidas tais variações no presente trabalho.

CONCLUSÃO

Nesta dissertação pode ser verificado que após a implementação do mecanismo de descoberta de topologia com o uso do algoritmo *Skeleton-tree* e a introdução das alterações propostas à técnica, as probabilidades de falhas apresentam resultados diferentes em alguns cenários simulados. No presente trabalho é apresentada uma alternativa ao método ativo da técnica de descoberta desenvolvida por Bejerano para o algoritmo *Skeleton-tree*, sendo utilizado o ARP como ferramenta de atualização dos dados das AFTs no lugar do ICMP. A alteração resulta em melhora na probabilidade de falha da descoberta de topologia em cenários com tempos de resposta menores (1 segundo). Ao considerar uma topologia de grande porte, com elevada quantidade de nós, esse cenário pode escalar melhor no processo de descoberta. Caso diversos blocos de endereços sejam segmentados para serem rasteados, o tempo cumulativo salvo em cada bloco resultaria em uma descoberta significativamente mais rápida. Dessa forma, a técnica de descoberta poderia ser adaptada a um sistema de monitoria, identificando entrada e saída de nós na rede e alterações na topologia. Essa abordagem, considerando a influência do tempo limite de espera na probabilidade de falha da descoberta de topologia de rede, não foi abordado originalmente por Yigal Bejerano em (BEJERANO, 2006) e (BEJERANO, 2009).

O ICMP possui a vantagem de poder ter seus pacotes gerados facilmente na rede com o comando *ping*, enquanto que o pacote ARP precisou ser construído na estação NMS para ser encaminhado no meio. Tradicionalmente o ARP é invocado por outro protocolo quando é necessário descobrir o endereço físico (MAC) de um endereço lógico conhecido. O ARP também é invocado pelo switch quando recebe um pacote com um endereço MAC destino que não consta na sua AFT.

Nos casos em que a informação dos endereços físicos de destino não se encontra disponível na tabela de encaminhamento, é necessária a geração de dados ARP para que a tabela seja atualizada. Ao eliminar o uso do ICMP e utilizando como método inicial diretamente o protocolo ARP em modo de difusão, os dados são atualizados sem necessidade de processamento do envido do ICMP na camada de rede (L3), eliminando o tempo gasto nessa etapa do processo. Em consequência também é eliminado um ponto de possível falha, seja por erros de comunicação na camada de rede ou por processamento que resultem em ultrapassar o tempo limite de espera. Dessa forma, foi possível descobrir a topologia de rede na grande maioria dos casos, localizando dispositivos L2 como os *switches* não gerenciáveis e *hubs*, por meio de coleta de dados obtidos da Bridge-MIB padrão. O interesse na descoberta, com pequeno atraso, dos nós adicionados à rede pode estar ligado a detecção de que um usuário que invada a rede. Um nó adicionando por um usuário à rede possibilitaria execução de programa malicioso, captura e inserção de pacotes, comprometendo dados sensíveis e a integridade da comunicação. A identificação

antecipada deste cenário permite aos administradores e sistemas restringirem a janela de atuação do invasor.

Direções futuras

Como direções futuras, destacam-se a extensão da descoberta de redes com implementação de diferentes VLANs. As informações do padrão 802.1q disponíveis na MIB IEEE8021-Q-BRIDGE-MIB poderiam ser analisadas em tentativa de serem usadas no processo de descoberta de VLAN. A adaptação do algoritmo *Skeleton-tree* nesse caso talvez não necessite grandes mudanças, se observado sua característica de receber redes distintas como dados de entrada. Todavia, a alteração na execução do processo de atualização das AFTs e coleta de dados precisa levar em consideração domínios de difusão L2 diferentes, definidos por cada VLAN. Os pacotes ARP gerados necessitaria de adaptação para as identificações (*tags*) de cada VLAN.

Novos experimentos podem ser realizados com o uso do ARP e tempos de espera com limites menores, em frações de 1 segundo, para analisar até que ponto é possível reduzir o tempo de espera limite e aumentar a velocidade de descoberta de topologia sem piorar a probabilidade de falha. Dessa forma, uma ferramenta de monitoria desenvolvida com a técnica poderia obter uma viso mais rápida da topologia de rede, identificando, de forma mais eficiente do que com a opção utilizando ICMP, falhas e dispositivos intrusos. Isso seria possível com a comparação de processos de descoberta de topologia executados de forma recorrente em uma mesma sub-rede. Também podem ser feitas avaliações em redes não simuladas como forma de consolidar os resultados obtidos neste trabalho.

Técnicas também podem ser desenvolvidas com a exploração de recursos presentes em outras formas de descoberta de topologia, analisando dados disponíveis em outras MIBs ou adicionando informações de outros protocolos. Desta forma poderia ser investigado a possibilidade de descobrir enlaces e caminhos além dos apresentados pela árvore do *spanning-tree*, como interfaces em estado de bloqueadas (*block*) pelo protocolo. Um estudo com a agregação de dados de MIBs que indiquem o estado de bloqueio e protocolos de vizinhança executados em L2 (LLDP, CDP) pode levar a identificação de novas conexões, levando a descoberta de topologia lógica mais próxima topologia física.

REFERÊNCIAS

BEJERANO, Y. Taking the skeletons out of the closets: A simple and efficient topology discovery scheme for large ethernet lans. In: IEEE. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings.* [S.l.], 2006. p. 1–13.

BEJERANO, Y. Taking the skeletons out of the closets: A simple and efficient topology discovery scheme for large ethernet lans. *IEEE/ACM Transactions on Networking*, v. 17, n. 5, p. 1385–1398, Oct 2009. ISSN 1063-6692.

BIERMAN, A.; JONES, K. S. *Physical Topology MIB*. RFC Editor, 2000. RFC 2922. (Request for Comments, 2922). Disponível em: ⟨https://rfc-editor.org/rfc/rfc2922.txt⟩.

CASE, J. D. et al. Simple Network Management Protocol (SNMP). *RFC 1157*, maio 1990. Disponível em: ⟨http://www.rfc-editor.org/rfc/rfc1157.txt⟩.

CHESHIRE, S. *IPv4 Address Conflict Detection*. [S.l.], 2008. Disponível em: ⟨http://www.rfc-editor.org/rfc/rfc5227.txt⟩.

CONGDON, P. Link layer discovery protocol. *RFC*, v. 2002, 2002.

GNS3, T. I. *General Architecture Documentation*. [S.l.], 2018. Https://gns3-server.readthedocs.io/en/latest/general.html.

GOVINDAN, R.; TANGMUNARUNKIT, H. Heuristics for internet map discovery. In: IEEE. *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. [S.l.], 2000. v. 3, p. 1371–1380.

HAGBERG, A. A.; SCHULT, D. A.; SWART, P. J. Exploring network structure, dynamics, and function using networkx. In: VAROQUAUX, G.; VAUGHT, T.; MILLMAN, J. (Ed.). *Proceedings of the 7th Python in Science Conference*. Pasadena, CA USA: [s.n.], 2008. p. 11 – 15.

ITU-T, I. IEC: Abstract Syntax Notation One (ASN. 1) Specification of Basic Notation. *Report no. ITU-T Rec. X*, v. 680, p. 8824–1, 2002.

MADHYASTHA, H. V. et al. iplane: An information plane for distributed services. In: USENIX ASSOCIATION. *Proceedings of the 7th symposium on Operating systems design and implementation*. [S.l.], 2006. p. 367–380.

MARCHETTA, P. et al. Don't trust traceroute (completely). In: ACM. *Proceedings of the 2013 workshop on Student workhop*. [S.l.], 2013. p. 5–8.

MCCLOGHRIE, K.; ROSE, M. T. Management Information Base for network management of TCP/IP-based internets. *RFC 1156*, maio 1990. Disponível em: ⟨http://www.rfc-editor.org/rfc/rfc1156.txt⟩.

MOTAMEDI, R.; REJAIE, R.; WILLINGER, W. A survey of techniques for internet topology discovery. *IEEE Communications Surveys & Tutorials*, IEEE, v. 17, n. 2, p. 1044–1065, 2015.

NORSETH, K.; BELL, E. *Definitions of Managed Objects for Bridges*. [S.l.], 2005. Disponível em: ⟨http://www.rfc-editor.org/rfc/rfc4188.txt⟩.

NOWICKI, K.; MALINOWSKI, A. Topology discovery of hierarchical ethernet lans without snmp support. In: IEEE. *Industrial Electronics Society, IECON 2015-41st Annual Conference of the IEEE*. [S.l.], 2015. p. 005439–005443.

PANSIOT, J.-J. et al. Extracting intra-domain topology from mrinfo probing. In: SPRINGER. *International Conference on Passive and Active Network Measurement*. [S.l.], 2010. p. 81–90.

POSTEL, J. Internet Control Message Protocol. *RFC 792*, set. 1981. Disponível em: ⟨http://www.rfc-editor.org/rfc/rfc792.txt⟩.

RODRIGUEZ, S. R. Topology discovery using cisco discovery protocol. *arXiv preprint arXiv:0907.2121*, 2009.

SCHAFER, I.; FELSER, M. Topology discovery in profinet. In: IEEE. *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*. [S.l.], 2007. p. 704–707.

SCHOLZ, T.; KRUSE, J. Tracing of hardware addresses in layer two bridged networks. In: IEEE. *Communications (ICC), 2015 IEEE International Conference on*. [S.l.], 2015. p. 1025–1030.

SPRING, N. et al. *How to resolve IP aliases*. [S.l.], 2004.

SZWARCFITER, J. *Teoria computacional de grafos: Os algoritmos*. Elsevier Editora Ltda., 2018. ISBN 9788535288858. Disponível em: ⟨https://books.google.com.br/books?id=OTBbDwAAQBAJ⟩.