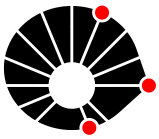

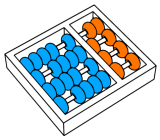

Monitoramento de infraestutura de rede e Análise de Causa Raiz

L. B. Alvetti

Relatório Técnico - IC-PFG-20-40
Projeto Final de Graduação
2020 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Monitoramento de infraestutura de rede e Análise de Causa Raiz

Leonardo Beretta Alvetti*

Resumo

Garantir alta disponibilidade e alta confiabilidade da infraestrutura de rede é algo de extrema importância para diversas empresas, especialmente para aquelas cujo negócio possui dependência de sistemas de informação. Mais do que isso, ter uma rápida resposta em caso de queda de conectividade em equipamentos é crucial para manter a continuidade da operação. Neste cenário, este trabalho se propõe a simular uma rede de equipamentos de telecomunicações, introduzir falhas de conectividade em alguns equipamentos, avaliar o impacto destas falhas nos demais equipamentos dessa rede e, com base nesses impactos e na topologia da rede, realizar a análise de causa raiz.

A análise causa raiz se propõe a estabelecer, em ordem decrescente de probabilidade, os equipamentos causadores da perda de conectividade da rede em situações nas quais vários equipamentos apresentam este sintoma.

1 Introdução

Atualmente, possuir uma área de tecnologia alinhada com o seu negócio, é fundamental para empresas, uma vez que sistemas de informação desempenham um papel crucial na maioria das operações [1]. Nesse cenário, garantir alta disponibilidade e alta confiabilidade destes mesmos sistemas, através do monitoramento da infraestrutura de rede, torna-se algo imperativo. Além disso, possuir um rápido tempo de resposta no caso de interrupções torna-se indispensável para garantir a continuidade do negócio e a produtividade das equipes.

Com a tendência de crescente complexidade da topologia, tamanho e criticidade das redes corporativas, detectar a origem do problema em caso de falhas desempenha um papel importante na atividade de *troubleshooting* [2] a fim de evitar que recursos computacionais e humanos sejam gastos na resolução de falhas que não são de fato a causa raiz do problema.

Neste trabalho, realizou-se um estudo sobre as redes corporativas, tendo como base a rede de um data center de uma grande instituição financeira brasileira, a fim de reproduzir a topologia de um data center, realizar a análise dessa rede, reproduzir falhas de perda de conectividade nos equipamentos de rede, inativando-o por completo ou apenas uma de suas interfaces, e a partir da análise dos impactos dessas falhas, identificar o equipamento que seja a causa raiz das falhas.

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.

2 Objetivo

O objetivo deste projeto é, a partir da simulação de uma rede corporativa, introduzir falhas de conectividade em determinados equipamentos - tanto falhas que inativam o equipamento por completo quanto falhas que inativam uma interface -, avaliar o comportamento da rede e realizar o processo de *Root Cause Analysis* [2] cujo objetivo é, em um cenário de falha em vários equipamentos da rede, estabelecer, em ordem decrescente de probabilidade, os possíveis equipamentos causadores dessa falha massiva.

3 Referencial Teórico

Topologia de redes corporativas

- Multi tier

Redes corporativas podem apresentar topologias muito diversas a fim de contemplar suas necessidades operacionais e requisitos de negócio. Neste trabalho, iremos focar em ambientes de Data Center, que são ambientes de rede que possuem uma topologia bem definida e requerem constante monitoração para garantir a alta disponibilidade da rede. Atualmente, o principal modelo de topologia encontrado em data centers é o modelo Multi-Tier (ou multi-camadas) [3]. Este modelo consiste em separar por camadas as funções de acesso, agregação e comutação entre sub-redes. Podemos observar um modelo multi-camadas na Figura 1.

Além disso, a separação em camadas permite que sejam definidas, mais facilmente, as políticas administrativas e de Qualidade de Serviço (QoS) mais adequadas a cada uma delas, ajudando assim também na manutenção e troubleshooting de problemas.

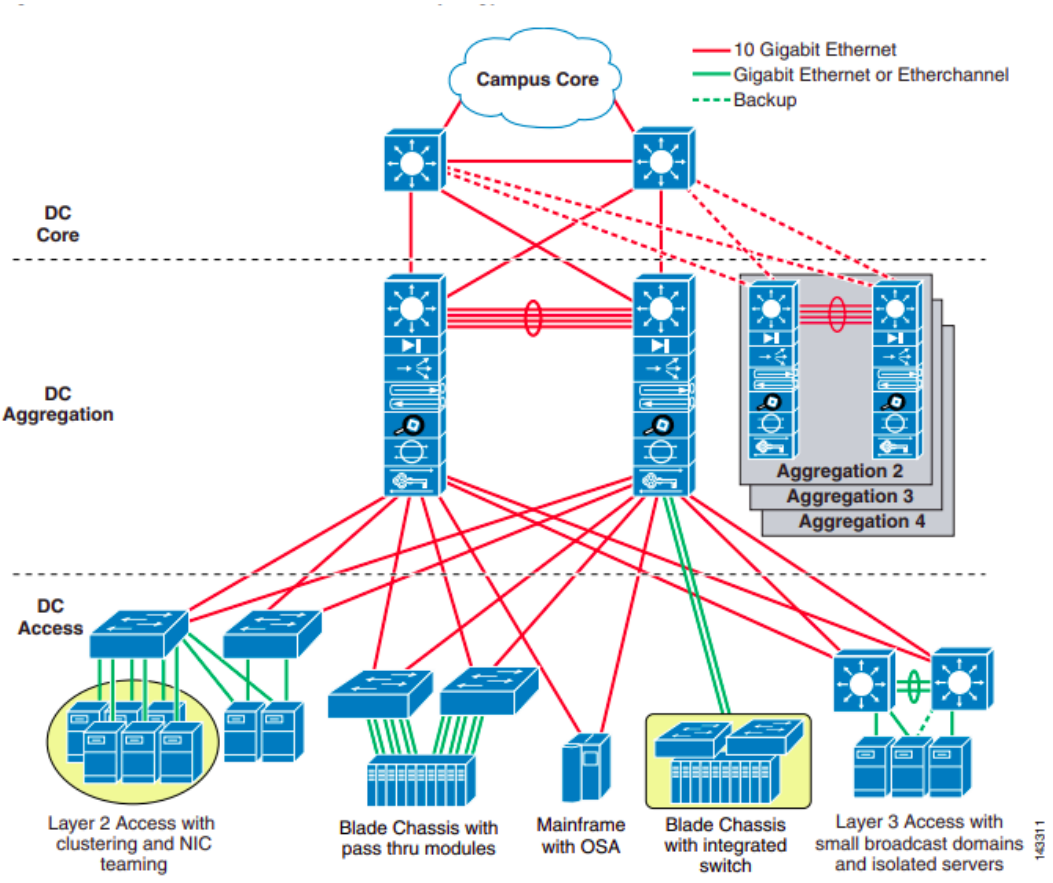

Figura 1: Ilustração do modelo de topologia *multi-tier*. Retirado de [3]

A camada *DC Core* tem como principal objetivo fornecer uma malha para comutação de pacotes com alta velocidade entre vários módulos de agregação. Essa camada também serve como gateway para a rede externa (na Figura 1 representada por *Campus Core*) onde outros módulos se conectam como, por exemplo, a extranet ou mesmo um equipamento de borda para Internet. Todos os enlaces que conectam o núcleo do data center normalmente usam interfaces com alta largura de banda para atender tanto ao alto número de requisições quanto aos requisitos de desempenho.

A camada *DC Aggregation Layer* é responsável por agregar as sessões que estão entrando e saindo do *Data Center*. Esta camada possui além de uma alta taxa de comutação de pacotes, uma alta de taxa de encaminhamento de pacotes. A camada de agregação também fornece outros serviços como balanceamento de carga do servidor, *firewall* e descarregamento de *Secure Sockets Layer (SSL)* para os servidores nos *switches* da camada de acesso.

Por fim, a camada **DC Access** é responsável por fornecer o acesso físico aos servidores, sendo também o primeiro ponto de *oversubscription* - que é o processo de atribuir uma certa largura de banda para um conjunto de usuários - dentro do Data Center.

- Spine Leaf

A topologia de rede leaf-spine [4] apresenta um recente crescimento e sua principal diferença, com relação à multi-tier, é o fato de que as três camadas - acesso, agregação e *core* - tem suas funções cumpridas por apenas duas camadas: spine e leaf, conforme Figura 2.

Figura 2: Ilustração do modelo de topologia *Spine-Leaf*. Retirado de [4]

Nesta arquitetura de duas camadas, cada *switch* de camada inferior, isto é: *leaf*, é conectado a cada um dos *switches* de camada superior, isto é: *spine*, em uma topologia de malha completa. A camada *leaf* consiste em *switches* de acesso que se conectam a dispositivos como servidores. A camada *spine* é o *backbone* da rede e é responsável por conectar todos os *switches* da camada *leaf*. Cada *switch* desta camada se conecta a cada *switch* da camada *spine*. O caminho é escolhido aleatoriamente para que a carga de tráfego seja distribuída uniformemente entre os *switches* da camada *spine*.

Dada essa topologia, não importa a qual *switch* da camada *leaf* um servidor está conectado, o número de saltos que um dado pacote enviado por um servidor terá de percorrer para chegar em outro servidor será o mesmo sempre, a menos que os servidores estejam na mesma *leaf* (folha). Essa abordagem mantém a latência em um nível previsível porque uma carga útil precisa apenas pular para um *switch spine* e outro *switch leaf* para chegar ao destino, favorecendo assim, o tráfego de pacotes lateralmente - o que explica o crescimento de sua adoção em *data centers* modernos por conta do maior uso de infraestrutura distribuída, tanto física quanto virtual.

GNS3

GNS3 é um software *open-source*, muito utilizado para virtualização de redes em geral [5]. Uma de suas grandes vantagens em comparação a outras ferramentas é o fato de que ele permite a virtualização de equipamentos Cisco, permitindo inclusive o acesso virtual à porta dos equipamentos, e consequentemente, acesso ao terminal de linha de comando desses equipamentos. O GNS3 também permite várias opções de customização dos equipamentos a serem simulados de uma maneira bem simples através de sua interface gráfica. Esta interface gráfica também permite uma fácil montagem da mesma.

Vale ressaltar que o GNS3 também permite que equipamentos virtualizados por ele também se conectem com equipamentos físicos. Além disso, o GNS3 possui uma grande comunidade ativa de usuários que pode ser bastante útil. Por fim, é importante notar que o GNS3 possui como desvantagem o alto consumo de CPU e memória, o que exige um bom hardware. Na Figura 3, podemos observar um exemplo de uma rede bastante simples simulada no GNS3.

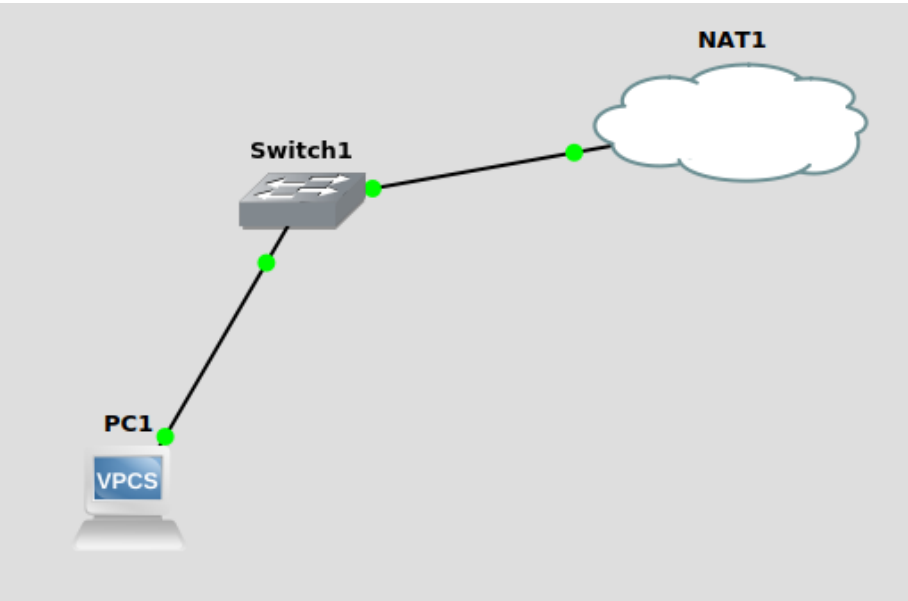

Figura 3: Exemplo de rede modelado no GNS3

Banco de dados em grafo

Representar os equipamentos e as conexões entre eles, e com isso a topologia da rede, em um banco de dados relacional se apresenta como um desafio muito maior do que em um banco de dados em grafo. Em situações nas quais as informações dos relacionamentos entre os elementos são importantes para o problema, como é o caso deste trabalho, bancos relacionais apresentam limitações que podem ser superadas com o uso de bancos de dados NoSQL. Na Figura 4, podemos ver um exemplo de grafo no Neo4j [7].

Em particular, bancos de dados em grafos armazenam os dados em vértices (ou nós) e arestas que representam suas relações, permitindo algumas vantagens sobre bancos de dados relacionais como flexibilidade e a velocidade de processamento de dados de relacionamento. Flexibilidade, pois um banco de dados em grafo não possui um *schema*, ou seja: não possui uma estrutura descrevendo quais os tipos de dados e como devem ser armazenados. A maior velocidade de processamento se dá pelo fato de que como os dados dos relacionamentos, representados pelas arestas, são armazenados juntamente com os vértices, acessar os dados desses relacionamentos possui o mesmo custo de acessar os dados dos vértices, ao passo que em um banco relacional, acessar dados de relacionamentos entre entidades é mais difícil uma vez que para acessar esses dados é necessário realizar *queries*, e em alguns casos *subqueries*, que utilizam *joins*, uma operação que é muito mais custosa. Além disso, a abordagem do problema orientada a grafos permite uma modelagem bastante fiel e intuitiva dada a natureza de uma rede de equipamentos.

Analisando uma comparação [6] entre as implementações de *Graph Database*, podemos observar que o Neo4j [7] é indicado como uma implementação atraente devido ao suporte nativo, para grafos, de propriedades, transações ACID, alta disponibilidade e rápido processamento de consultas. Na figura 4 podemos ver um exemplo de grafo no Neo4j.

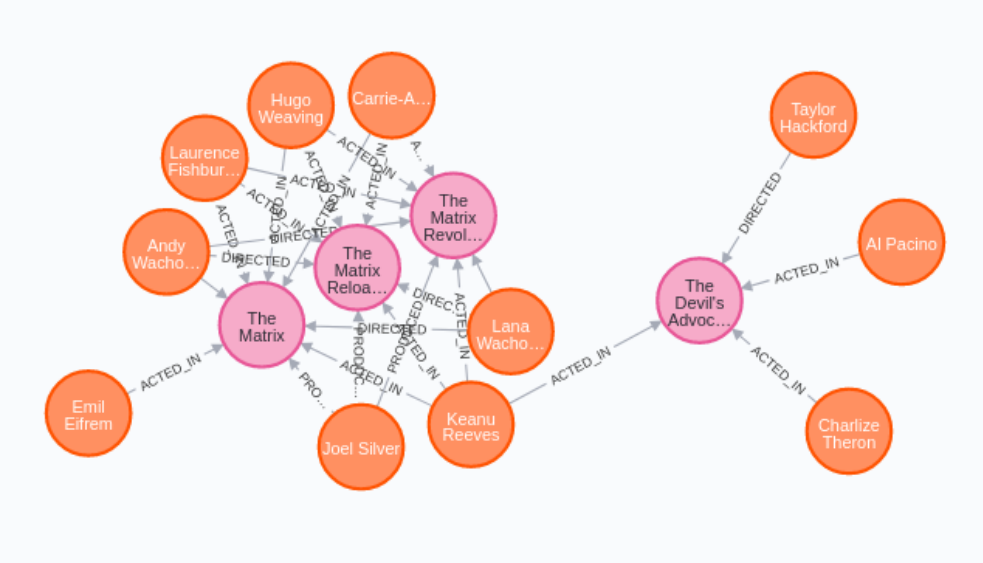

Figura 4: Exemplo de um grafo no Neo4j

Conforme dito anteriormente, enquanto bancos de dados relacionais recuperam dados interconectados, isto é: dados referentes aos relacionamentos entre entidades, por meio de operações mais custosas computacionalmente, as implementações em grafo resolvem esse problema por meio de operações nativas de travessia entre os vértices e arestas dos grafos. Para realizar essas operações de busca em grafos, existem algumas linguagens disponíveis para o Neo4j e cada uma delas possui propriedades diferentes. Neste trabalho optou-se pela linguagem Cypher [8], que é semelhante à linguagem MySQL e foi projetada para ser amigável ao desenvolvedor. As consultas realizadas em Cypher podem ser escritas para recuperar dados de acordo com certos padrões, permitindo assim uma maior autonomia e flexibilidade de trabalho para implementação de *Root Cause Analysis*. Para ilustrar isso, temos um pequeno exemplo de uma consulta para contar todas as conexões de um vértice:

Na consulta acima, utilizamos a linguagem Cypher para navegar entre os vértices e arestas do grafo procurando o grafo que cuja *property* "Name" corresponde a "Tom Hanks". Em seguida, são retornados os vértices que se encaixem no padrão de relação entre vértices especificado, no caso filmes em que "Tom Hanks" atuou; podemos observar a resposta desta consulta na Figura 5.

Figura 5: Grafo do Neo4j recuperado através de uma consulta em Cypher

4 Metodologia

Inicialmente, os modelos lógicos de topologia *Multi-tier* e *Spine-Leaf* foram simulados no GSN3, conforme Figuras 6 e 7, respectivamente. Inicialmente, este trabalho se propunha a explorar a análise de causa raiz tanto para perdas de conectividade de equipamentos inteiros quanto de interface. Porém, pela falta de poder computacional, foi necessário realizar a troca de *routers* por *switches* pois, caso isso não fosse feito, não seria possível simular uma rede. Além disso, como este trabalho se propõe a explorar a análise de causa raiz apenas para perda de conectividade, foram excluídos *firewalls* de cada topologia.

Além disso, os servidores tiveram de ser substituídos por *Virtual Personal Computers* (VPCs) pelo mesmo motivo acima, mas vale ressaltar que esta troca não apresenta impactos para análise de causa raiz uma vez que tanto servidores quanto VPCs são as folhas da rede e a conectividade de outros equipamentos em uma rede não depende das folhas.

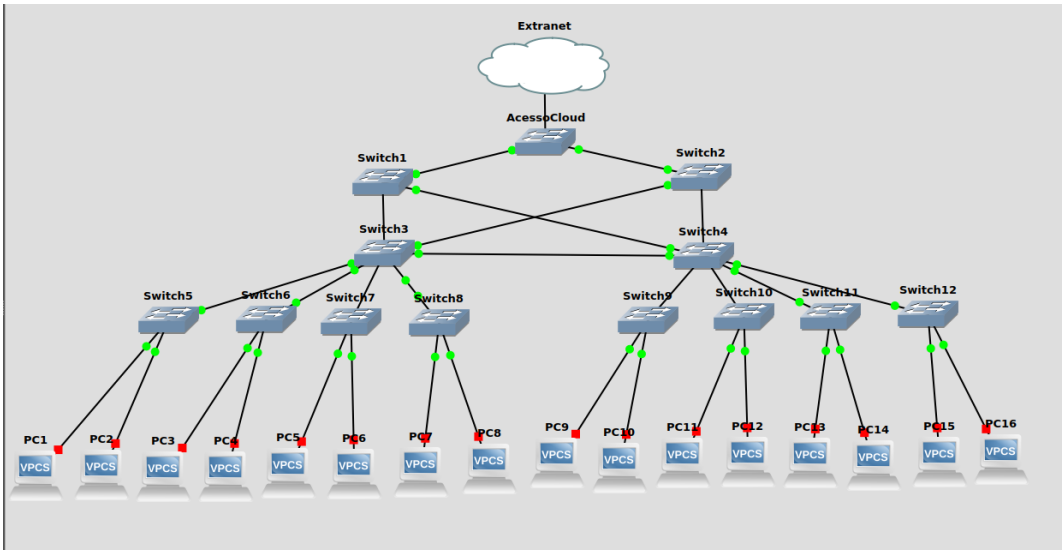

Figura 6: Topologia *Multi-tier* modelada no GNS3

Em seguida, ambos os modelos foram traduzidos para o Neo4j pois, conforme mencionado anteriormente, a natureza de uma rede de equipamentos de telecomunicações se assemelha a um grafo e o Neo4j provê ferramentas para a realização de consultas que serão utilizadas na análise de causa raiz (*Root Cause Analysis*). Podemos observar as correspondências entre os modelos topológicos e sua modelagem no Neo4j nas Figuras 10 e 11.

Cada equipamento, representado no grafo como um vértice, foi modelado com suas conexões, representadas por arestas, e foram inseridos *properties* - que podem ser entendidas como colunas de um banco relacional - que representam propriedades do equipamento. Os *labels* inseridos foram: *"IP"*, *"Hostname"*, *"ID"*, *"Status"* e *"Dependencias"*. Na Tabela 1 estão listados as *properties* e a descrição do que cada um deles.

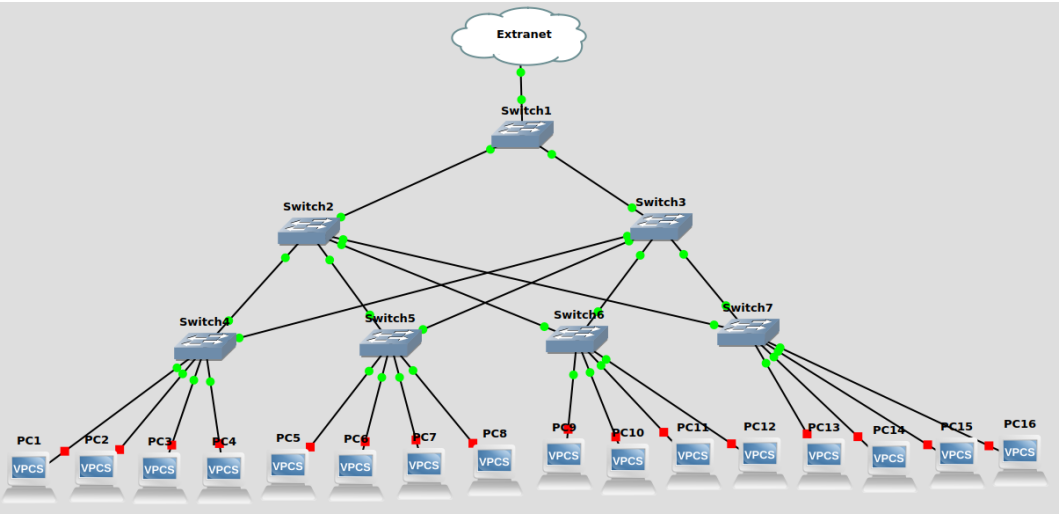

Figura 7: Topologia *Spine-Leaf* modelado no GNS3

Label	Descrição
IP	Endereço IP do equipamento
Hostname	Hostname do equipamento
ID	Número identificador único do equipamento
Status	Status de conectividade do equipamento - Connected/Disconnected
Dependencia	Lista todos os equipamentos que provém conectividade

Tabela 1: Propriedades dos equipamentos


```
{
  "identity": 35,
  "labels": [
    "router"
  ],
  "properties": {
"hostname": "r5 ",
"dependencias": "r2 ,r3 ",
"id": "105",
"ip": "10.238.70.105",
"status": "Connected"
  }
}
```


Figura 8: Propriedades de um equipamento modelado no Neo4j

Como podemos observar na Figura 8, o equipamento de ID "105" possui o *IP* "10.238.70.105", *hostname* "r5", *Status "Connected"* e apresenta dependência de conectividade com os equipamentos de hostname "r2" e "r3". Neste exemplo, como a conectividade do equipamento "r5" é provida pelos equipamentos "r2" e "r3", podemos dizer estes últimos são os equipamentos "pais" de "r5". Tendo os equipamentos modelados dentro do Neo4j e com a rede sendo simulada, foram introduzidas falhas em determinados equipamentos, isto é: quedas, para que a perda de conectividade de diversos equipamentos ao mesmo tempo fosse estudada. Os testes seguiram os seguintes passos:

1. Introduzir a queda em um ou mais equipamentos da rede e observar a perda ou não de conectividade dos demais equipamentos no GNS3.

2. Exportar essas quedas para o grafo. A representação de um equipamento sem conectividade no grafo será feita através da mudança da property Status de *"Connected"* para *"Disconnected"*.

3. Execução de consultas através de *queries* no grafo para avaliar as dependências de cada equipamento e assim determinar o causador primário da queda de conexão na rede. Os passos para realizar essa análise são os seguintes:

   (a) Determinar todos os equipamentos com Status igual a *"Disconnected"*;

   (b) Gerar duas listas iguais com todos esses equipamentos. Uma delas será usada como base de consulta dos equipamentos sem conectividade e a outra será a lista de equipamentos candidatos a serem causa raiz. Nesta segunda lista ocorrerão as interações que estão listadas nos passos abaixo e, ao fim da análise, os equipamentos que restarem nela deverão, em teoria, ser os equipamentos causadores da perda de conectividade em grande parte da rede;

   (c) Para cada equipamento da lista gerada no item 3.b, realizar uma consulta e ler a *property* "Dependencias";

   (d) Gerar um dicionário cujas tuplas terão como chave o *hostname* de um equipamento sem conectividade e como valor uma lista com os *hostnames* dos equipamentos "pais";

   (e) Para cada equipamento "pai", consultar a *property Status*;

   (f) Gerar um dicionário cujas tuplas terão como chave o *hostname* de um equipamento "pai" e como valor a resposta da consulta do item 3.e;

   (g) Verificar se ao menos um dos equipamentos listados na propriedade "Dependencias"está na lista de equipamentos sem conexão. Caso exista pelo menos um equipamento que cumpre essa condição, podemos inferir que o equipamento em questão não é o causador primário da queda. Caso todos os equipamentos listados na propriedade "Dependencias" tenham conectividade, o equipamento permanece na lista como candidato;

   (h) Remover da lista de possíveis candidatos os equipamentos que cumprirem a segunda condição do passo 3.g, isto é: remover aqueles que possuem dependência de outros equipamentos e os mesmos se encontram sem conectividade;

(i) Por fim, como existem situações em que mais de um equipamento pode ser a causa raiz e eles podem estar na mesma camada ou possuir uma ligação direta, será feita uma consulta no grafo a fim de estabelecer a distância dos demais equipamentos sem conectividade para o equipamento apontado como causa raiz.

Os passos 1 e 2 foram realizados manualmente, via GNS3 e Neo4j respectivamente, enquanto o passo 3 foi realizado via *script* em Python 3. A opção pelo script se deu pelo fato de que uma rede corporativa pode ter várias dezenas de equipamentos - em alguns casos, centenas ou mesmo, em raros casos, milhares -, o que tornaria essa análise inviável em uma situação real. Entretanto, antes de que os testes fossem conduzidos com as topologias propostas, foi construída uma pequena prova de conceito a fim de validar a efetividade do pseudo-algoritmo proposto no passo 3. Essa prova foi baseada em um grafo representando uma rede qualquer, com conexões redundantes e com catorze equipamentos, sendo que dois deles eram a 'origem' e 'destino' - podemos interpretar origem como rede externa e destino um servidor qualquer - e doze equipamentos de rede de fato. Podemos observar a topologia da rede utilizada como prova de conceito na Figura 9.

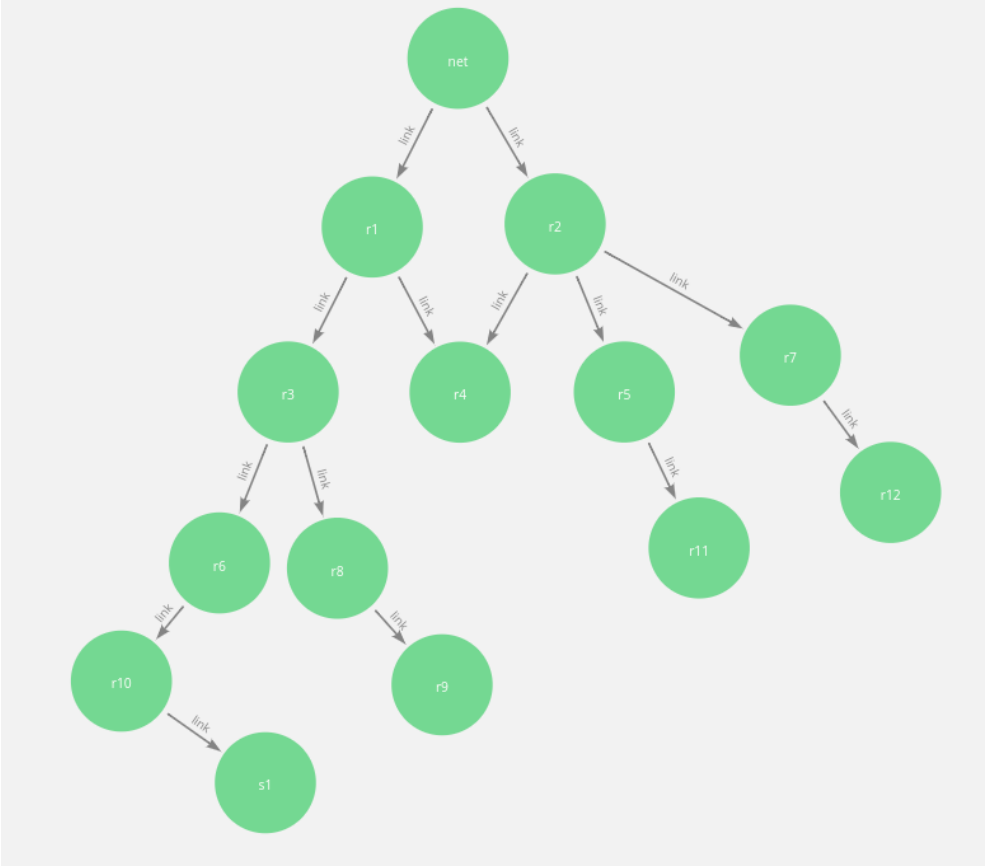

Figura 9: Topologia utilizada para PoC do algoritmo descrito no item 3 no Neo4j

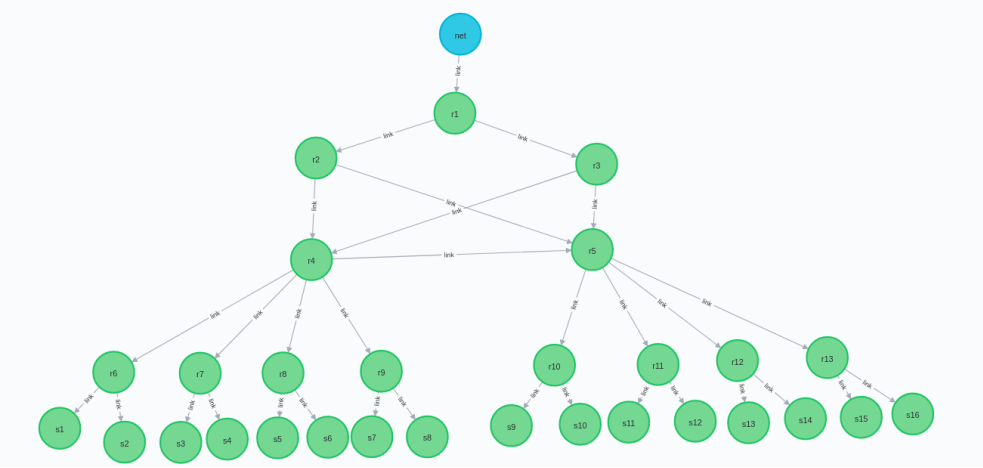

Figura 10: Topologia *mutl-tier* modulada no Neo4j

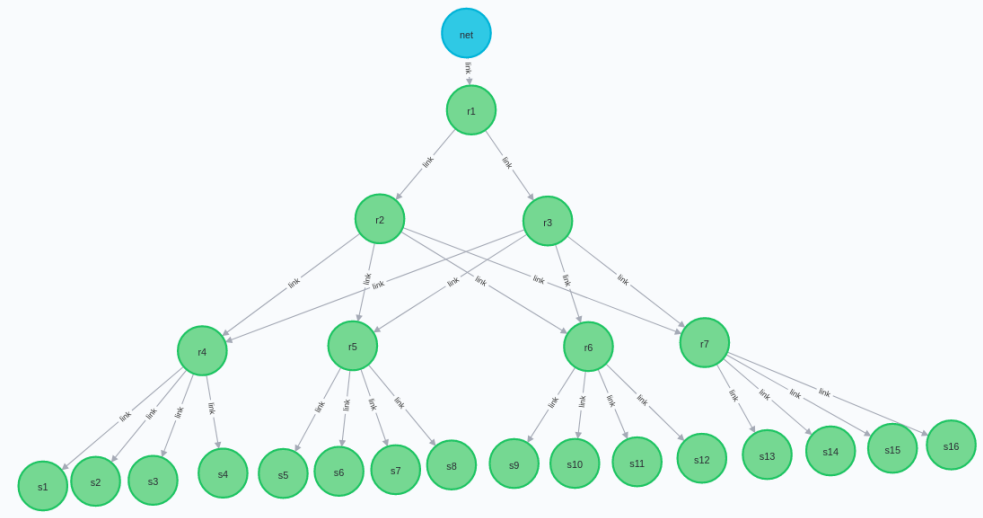

Figura 11: Topologia *spine-leaf* modulada no Neo4j

O desenvolvimento do código em Python 3 utilizou a biblioteca "neo4j"para realizar a conexão com o banco de dados em grafo do Neo4j e, com isso, consultar as informações de interesse. Após realizar a conexão com o banco de dados no Neo4j, o script realiza a consulta descrita no passo 3.a e ela retorna todos os equipamentos sem conectividade bem como os equipamentos "pais" de cada.

Com esses dados são construídas as estruturas que permitirão a análise de causa raiz, que são: a lista de todos os equipamentos sem conectividade (esta lista será usada como referência), a lista de todos os candidatos a serem causa raiz (esta lista é preenchida inicialmente com todos os equipamentos sem conectividade). Em seguida, para cada equipamento com *status "Disconnected"*, os equipamentos "pais" são inseridos em um dicionário, conforme passo 3.d; neste dicionário *key* é o *hostname* de um equipamento com *status "Disconnected"* e *value* é uma lista com todos os equipamentos "pais". Após isso, é realizada uma consulta em cada equipamento "pai", obtendo o *status* do mesmo. Esse *status* é inserido, como *value*, em um segunda dicionário (passos 3.e e 3.f).

Como dito anteriormente no passo 3.g, se um equipamento está sem conectividade e seus pais também estão, este equipamento é excluído da lista de possíveis causadores. Caso pelo menos um de seus equipamentos "pais" possua conectividade, ele permanece na lista de possíveis candidatos. Ao final desse processo com todos os equipamentos, restarão na lista de possíveis candidatos apenas os equipamentos sem conectividade (*status "Disconnected"*) cujos equipamentos "pais" estão com conectividade (*status "Connected"*). Essa abordagem é interessante pois caso existam equipamentos distintos causando perda de conectividade em pontos diferentes da rede, ambos serão listados como causa raiz do problema.

Entretanto, como descrito no passo 3.i, um equipamento pode estar com problemas de conectividade de fato e seus equipamentos "pai" também. Para contornar essa situação, será gerada uma lista, em ordem de probabilidade, dos equipamentos que também podem estar com problemas de conectividade, ou seja: equipamentos cujos problemas de conectividade são reais, porém encontram-se disfarçados pelos problemas de conectividade dos equipamentos "pais". A probabilidade será gerada através de uma consulta no Neo4j na qual será calculada, para cada equipamento sem conectividade que foi descartado como causa raiz, a distância entre ele e os equipamentos que foram considerados causa raiz. Caso exista mais de um equipamento identificado como causa raiz, será considerada a distância de menor valor. O cálculo destas distâncias será feito utilizando uma chamada via *script* Python; essa chamada irá invocar o método *"shortest_path"* [9] do Neo4j, que retorna a menor distância entre dois vértices do grafo. Utilizando a topologia da Figura 9, podemos ver na Figura 12 o retorno da consulta de *shortest_path* entre os equipamentos r1 e r9.

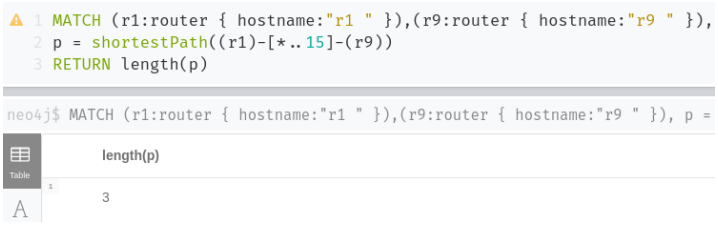

Figura 12: Consulta de *shortest_path* no Neo4j

Os cenários de testes iniciais são situações de queda de um único equipamento em cada uma das camadas de cada topologia alvo de estudo e estão descritos na tabela 2.

Nome do cenário	Topologia	Equipamentos em queda
Cenário 1 - Queda Equipamento *Core*	*Multi-tier*	R2
Cenário 2 - Queda Equipamento *Aggregation*	*Multi-tier*	R5
Cenário 3 - Queda Equipamento *Access*	*Multi-tier*	R7
Cenário 4 - Queda Equipamento *Spine*	*Spine-Leaf*	R2
Cenário 5 - Queda Equipamento *Leaf*	*Spine-Leaf*	R5

Tabela 2: Cenários iniciais de testes

Em seguida, foram introduzidas quedas em mais de uma camada. Estes cenários estão listados na Tabela 3.

Nome do cenário	Topologia	Equipamentos em queda
Cenário 6 - Queda Equips *Core* e *Aggregation*	*Multi-tier*	R2, R5
Cenário 7 - Queda Equips *Aggregation* e *Access*	*Multi-tier*	R4, R6
Cenário 8 - Queda Equips *Core* e *Access*	*Multi-tier*	R3, R7
Cenário 9 - Queda Equips *Spine* e *Leaf*	*Spine-Leaf*	R3, R4

Tabela 3: Cenários de testes envolvendo mais de uma camada

Os últimos cenários de testes estão descritos na Tabela 4 e contemplam perda de conectividade em interfaces e não no equipamento todo.

Nome do cenário	Topologia	Equipamentos
Cenário 10 - Queda interface *Core* e *Aggregation*	*Multi-tier*	Interface R2, R5
Cenário 11 - Queda interface *Aggregation* e *Access*	*Multi-tier*	Interface R4, R9
Cenário 12 - Queda interface *Extranet* e *Core*	*Multi-tier*	Interface R1, R3
Cenário 13 - Queda interface *Spine* e *Leaf*	*Spine-Leaf*	Interface R3, R4
Cenário 14 - Queda interface *Extranet* e *Spine*	*Spine-Leaf*	Interface R1, R2

Tabela 4: Cenários de testes envolvendo perda de conectividade em interfaces

5 Resultados e Discussão

A topologia utilizada para a prova de conceito do algoritmo de análise de causa raiz pode ser observada na Figura 13. Os vértices em vermelho estão com o *Status "Disconnected"* e não apresentam conectividade. Nesta situação, na qual não havia apenas um equipamento causador da falha de conectividade, a análise encontrou como causadores de falha de conectividade os equipamentos r1 e r5, conforme o esperado. Os resultados podem ser observados nas figuras 13 e 14.

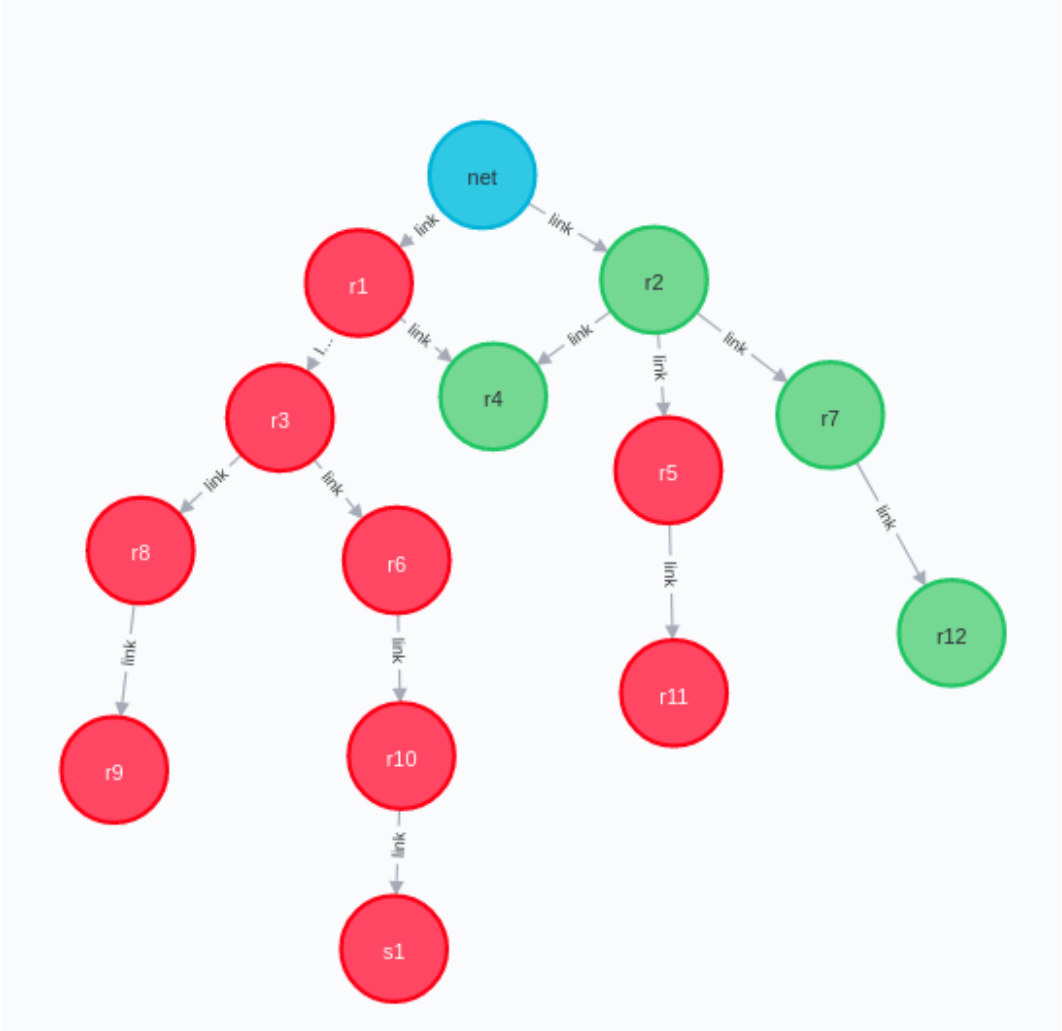

Figura 13: Topologia da prova de conceito com quedas

Com o sucesso da prova de conceito do algoritmo, foram iniciados os testes nas topologias alvo de estudo. No cenário 1, Figuras 15 a 17, foi introduzida uma falha de conectividade, no caso uma queda do equipamento, apenas no equipamento de hostname e conforme podemos observar na Figura 17 que mostra o output do script que realiza a análise, a causa

```
leo@leo-X510UR:~/Documents$ sudo python3 rootcause.py
Os equipamentos causa raiz da queda da rede:  ['r1 ', 'r5 ']
Os equipamentos que podem também apresentar problemas são:  ['r3 ', 'r11', 'r6 ', 'r8 ', 'r10']
leo@leo-X510UR:~/Documents$
```

Figura 14: Resultado da análise de causa raiz para a rede da imagem 13

primária foi apontada corretamente e os equipamentos que poderiam também apresentar um problema real foram listados em ordem decrescente de possibilidade, isto é: do mais próximo ao equipamento apontado como causa raiz para o mais distante. Os resultados obtidos nos cenários 2, 3, 4 e 5 foram os mesmos, conforme Figuras de 18 a 29.

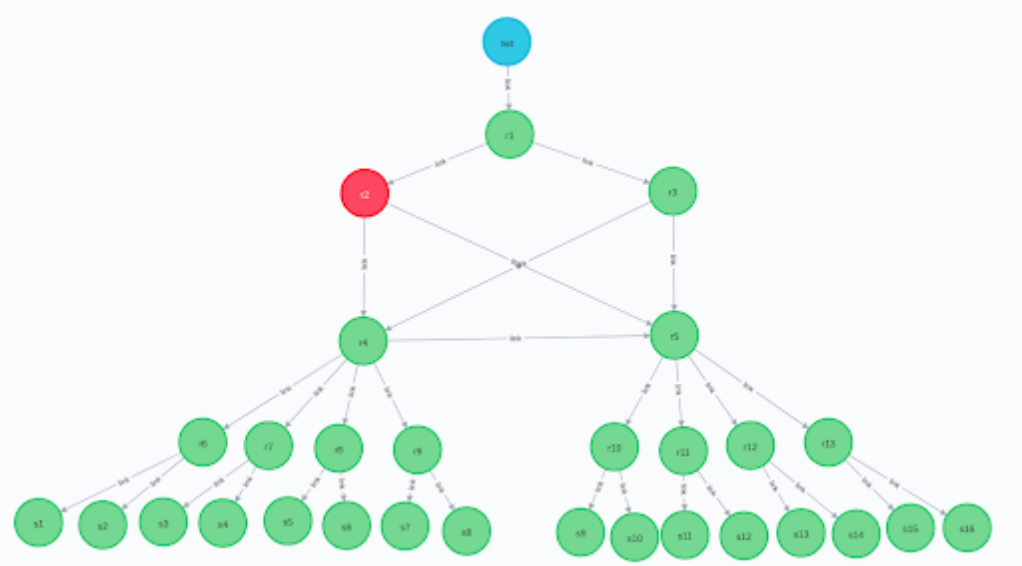

Figura 15: Equipamentos com falha do Cenário 1

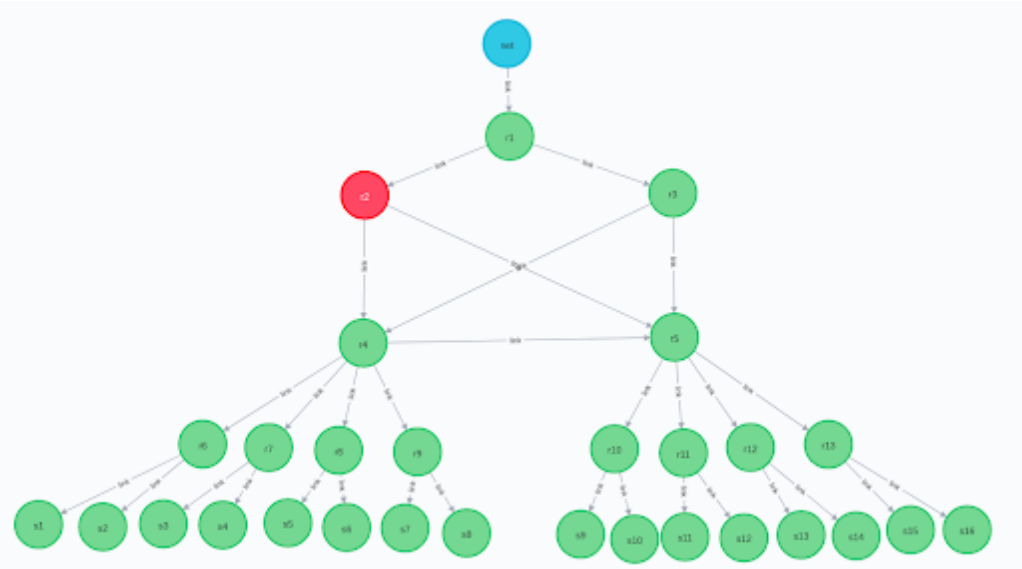

Figura 16: Comportamento da rede com base no Cenário 1

```
leo@leo-X510UR:~/Documents$ sudo python3 rootcause.py
Os equipamentos causa raiz da queda da rede:  ['r2 ']
Equipamentos secundarios:  []
leo@leo-X510UR:~/Documents$
```

Figura 17: Resultado da RCA para o Cenário 1

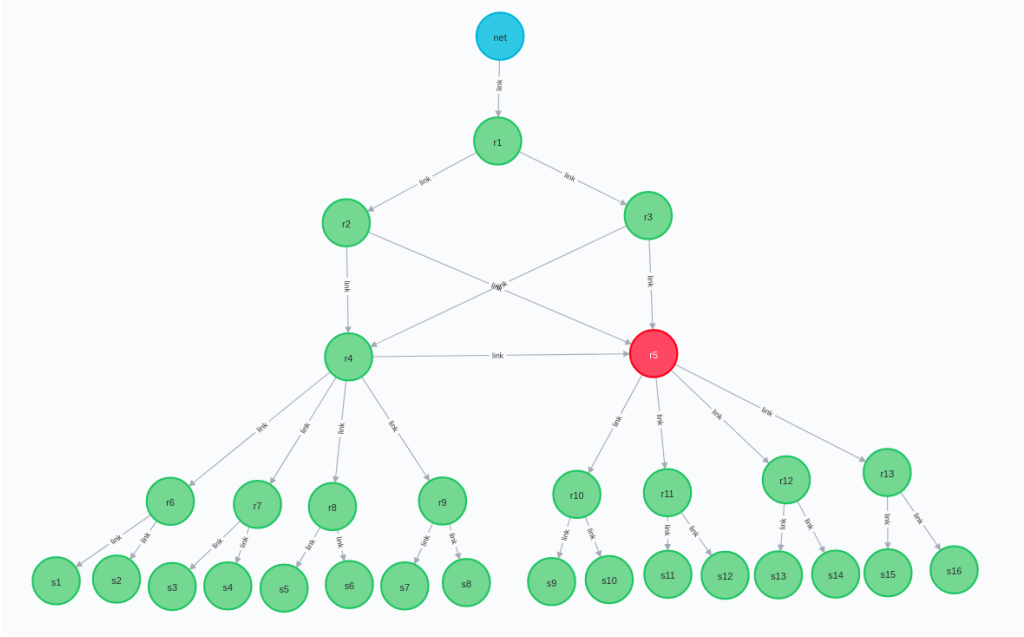

Figura 18: Equipamentos com falha do Cenário 2

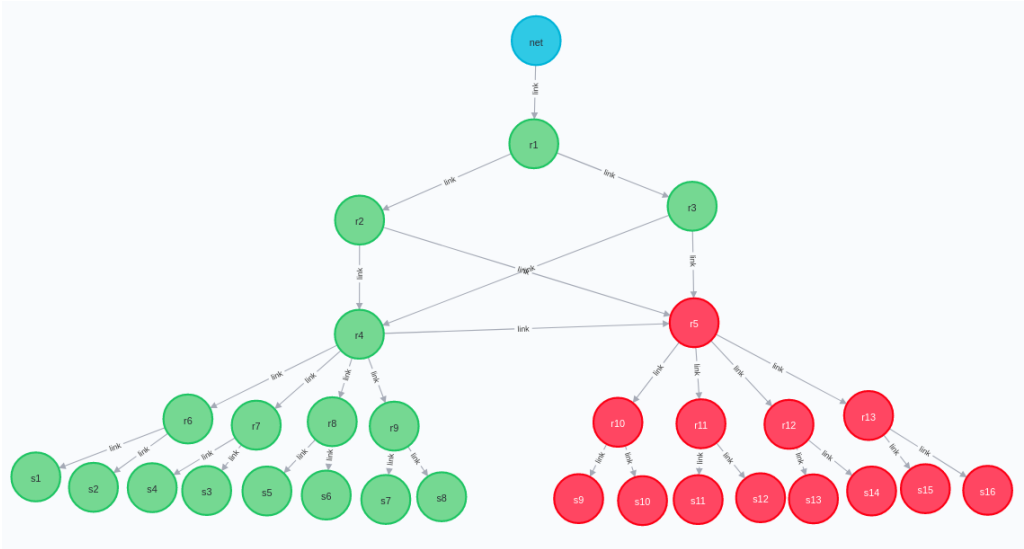

Figura 19: Comportamento da rede com base no Cenário 2

```
leo@leo-X510UR:~/Documents$ sudo python3 rootcause.py
Os equipamentos causa raiz da queda da rede:  ['r9 ', 'r10', 'r11', 'r12']
Equipamentos secundarios:  ['s9 ', 's10', 's11', 's12', 's13', 's14', 's15', 's16']
leo@leo-X510UR:~/Documents$
```

Figura 20: Resultado da RCA para o Cenário 2

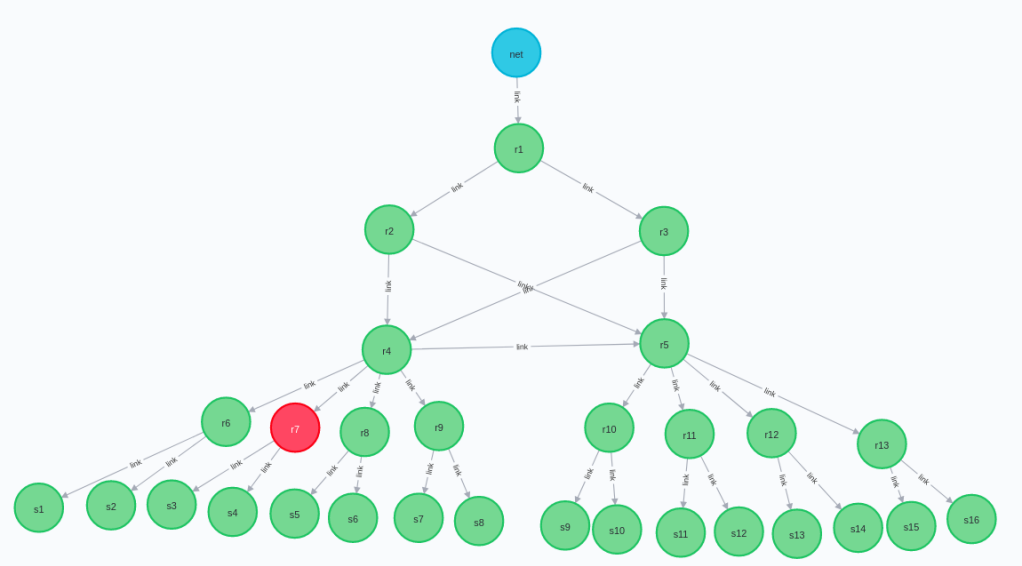

Figura 21: Equipamentos com falha do Cenário 3

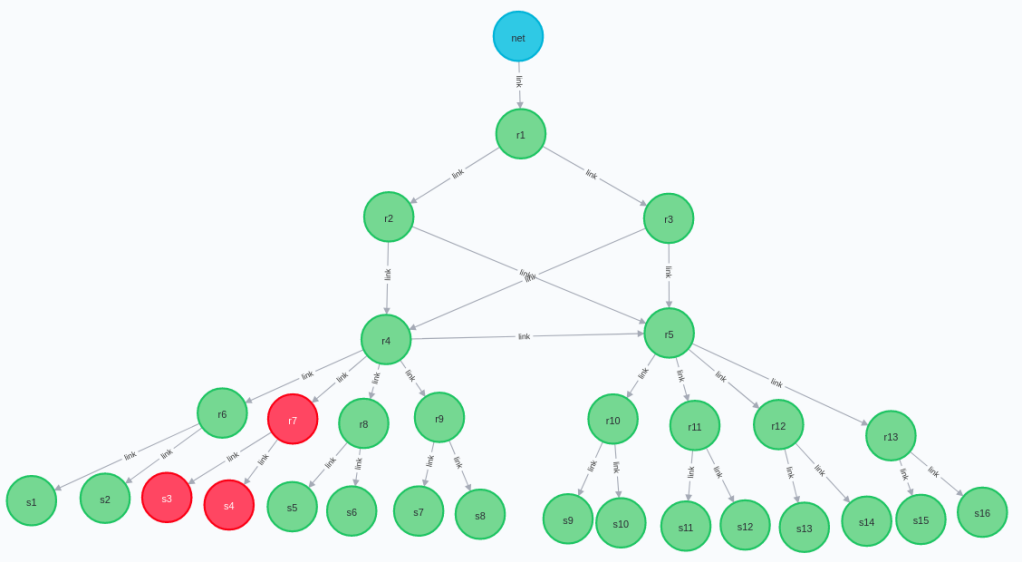

Figura 22: Comportamento da rede com base no Cenário 3

```
leo@leo-X510UR:~/Documents$ sudo python3 rootcause.py
Os equipamentos causa raiz da queda da rede:  ['r7 ']
Equipamentos secundarios:  ['s3 ', 's4 ']
leo@leo-X510UR:~/Documents$
```

Figura 23: Resultado da RCA para o Cenário 3

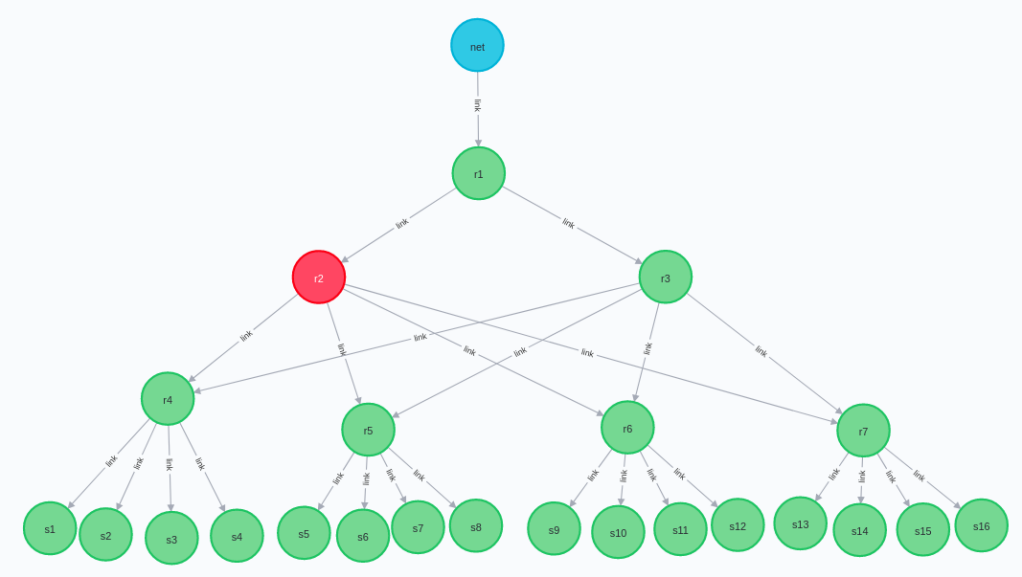

Figura 24: Equipamentos com falha do Cenário 4

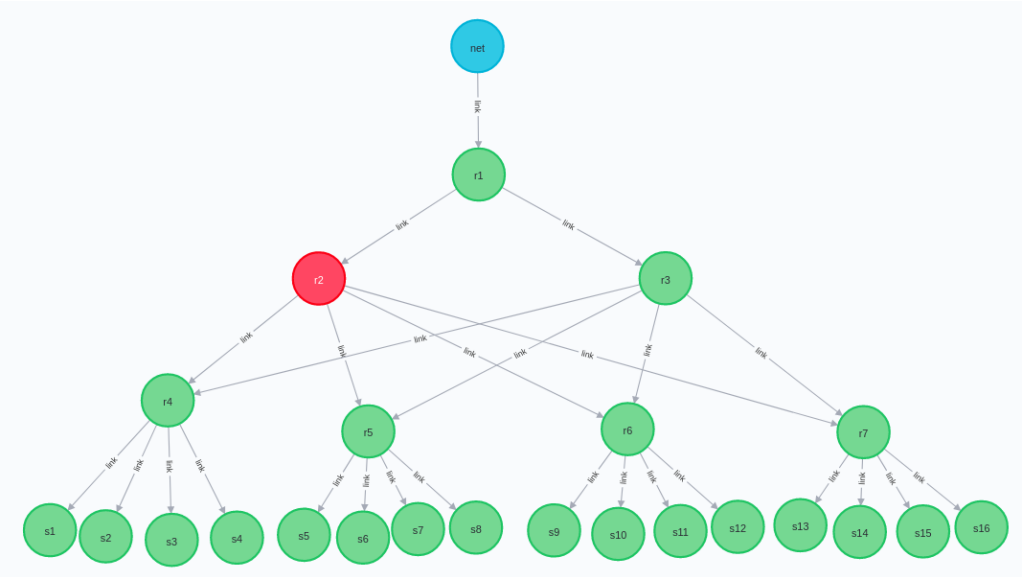

Figura 25: Comportamento da rede com base no Cenário 4

```
leo@leo-X510UR:~/Documents$ sudo python3 rootcause.py
Os equipamentos causa raiz da queda da rede:  ['r2 ']
Equipamentos secundarios:  []
leo@leo-X510UR:~/Documents$
```

Figura 26: Resultado da RCA para o Cenário 4

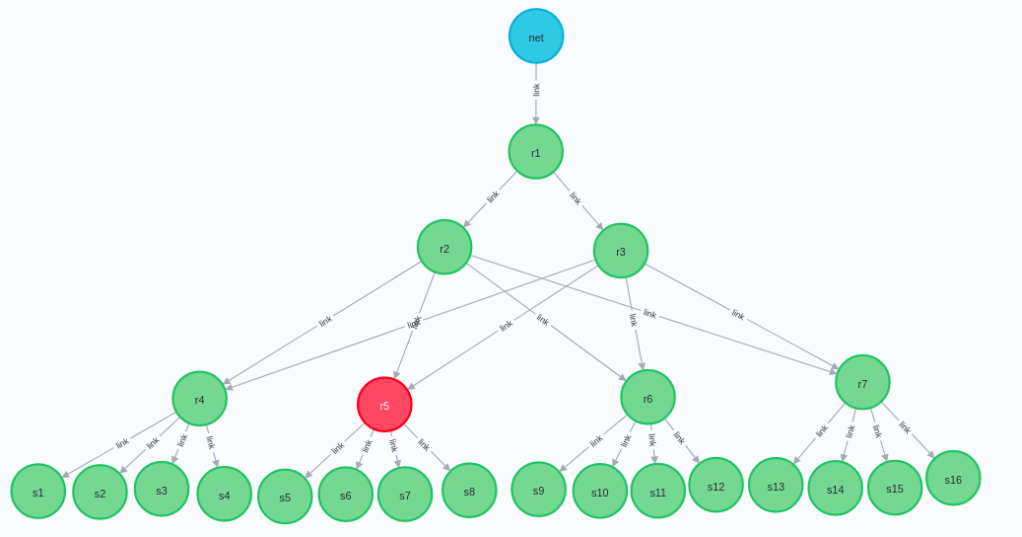

Figura 27: Equipamentos com falha do Cenário 5

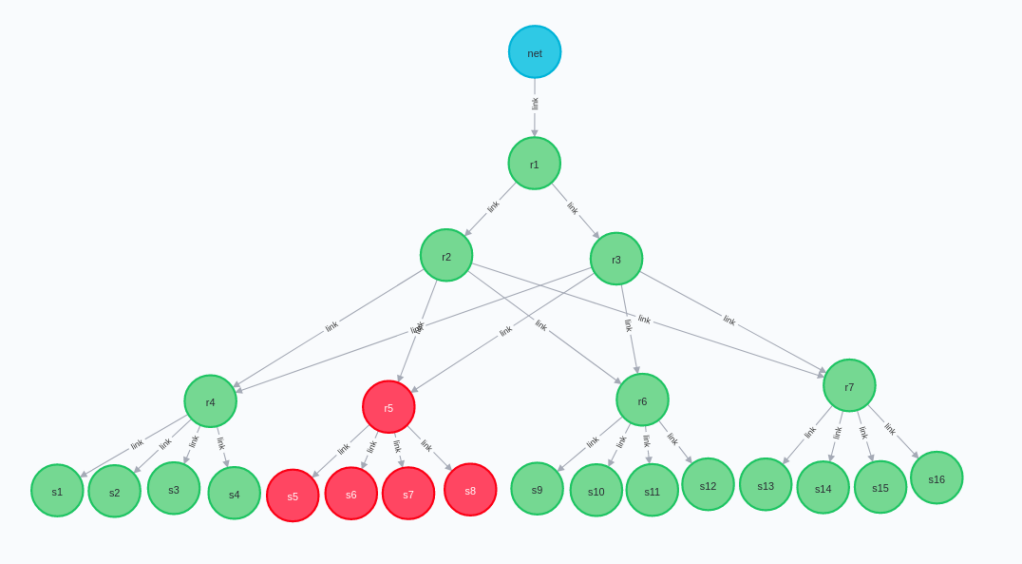

Figura 28: Comportamento da rede com base no Cenário 5

```
leo@leo-X510UR:~/Documents$ sudo python3 rootcause.py
Os equipamentos causa raiz da queda da rede:  ['r5 ']
Equipamentos secundarios:  ['s5 ', 's6 ', 's7 ', 's8 ']
leo@leo-X510UR:~/Documents$
```

Figura 29: Resultado da RCA para o Cenário 5

Para os cenários descritos na Tabela 3, cenário 6 a cenário 9, foram introduzidas falhas de quedas simultâneas em equipamentos de camadas diferentes. Conforme Figuras 30 a 32 e 36 a 38, correspondentes aos cenários 6 e 8 respectivamente, a análise de causa raiz se comportou como esperado, apontando corretamente os equipamentos com falhas. No entanto, as Figuras 33 a 35, correspondentes ao cenário 7, mostram que apenas o R4 foi listado como causa raiz e não R4 e R7. A análise conclui que R7 não poderia ser apontado como causa raiz pelo fato de que o equipamento responsável pela sua conectividade, que no caso é R4, não apresentava conectividade e, por este motivo, R7 foi descartado como candidato a causa raiz. Entretanto, vale observar que R7 é listado como o equipamento mais provável de apresentar problemas, conforme imagem 35. No cenário 9, o único envolvendo perda de conectividade de dois equipamentos na topologia *Spine-Leaf*, a análise se comportou conforme esperado. Além disso, podemos observar que por conta da topologia *Spine-Leaf* a situação ocorrida no cenário 7 não pode ser reproduzida.

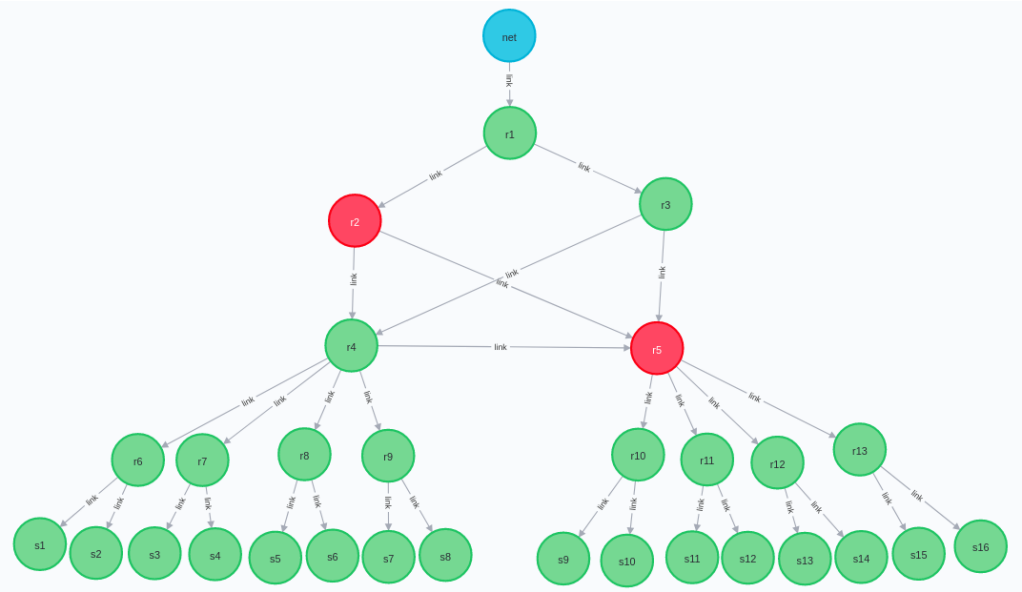

Figura 30: Equipamentos com falha do Cenário 6

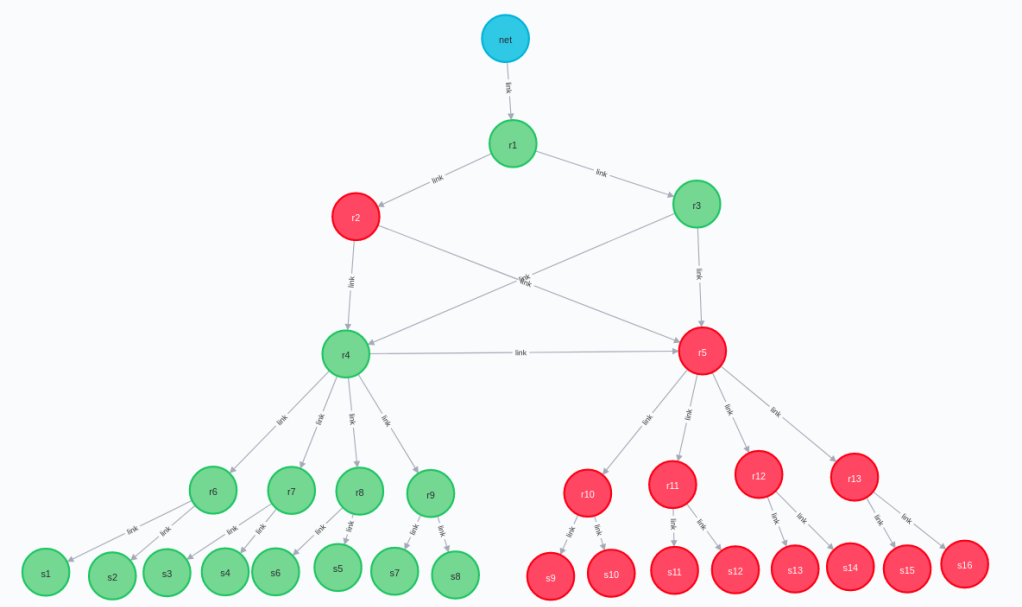

Figura 31: Comportamento da rede com base no Cenário 6

```
leo@leo-X510UR:~/Documents$ sudo python3 rootcause.py
Os equipamentos causa raiz da queda da rede:  ['r2 ', 'r5 ']
Equipamentos secundarios:  ['r10', 'r11', 'r12', 'r13']
                           ['s9 ', 's10', 's11', 's12', 's13', 's14', 's15', 's16']
leo@leo-X510UR:~/Documents$
```

Figura 32: Resultado da RCA para o Cenário 6

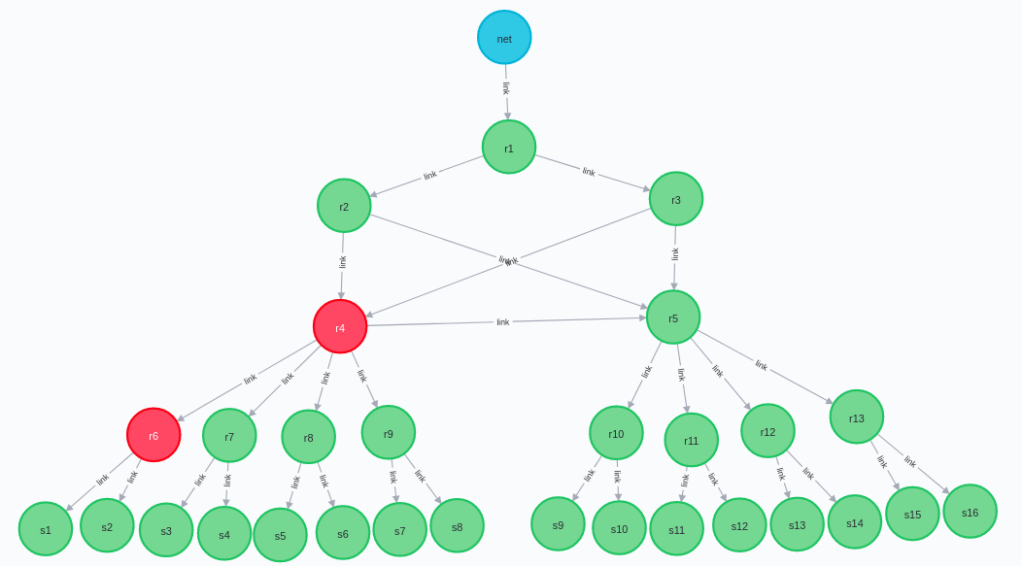

Figura 33: Equipamentos com falha do Cenário 7

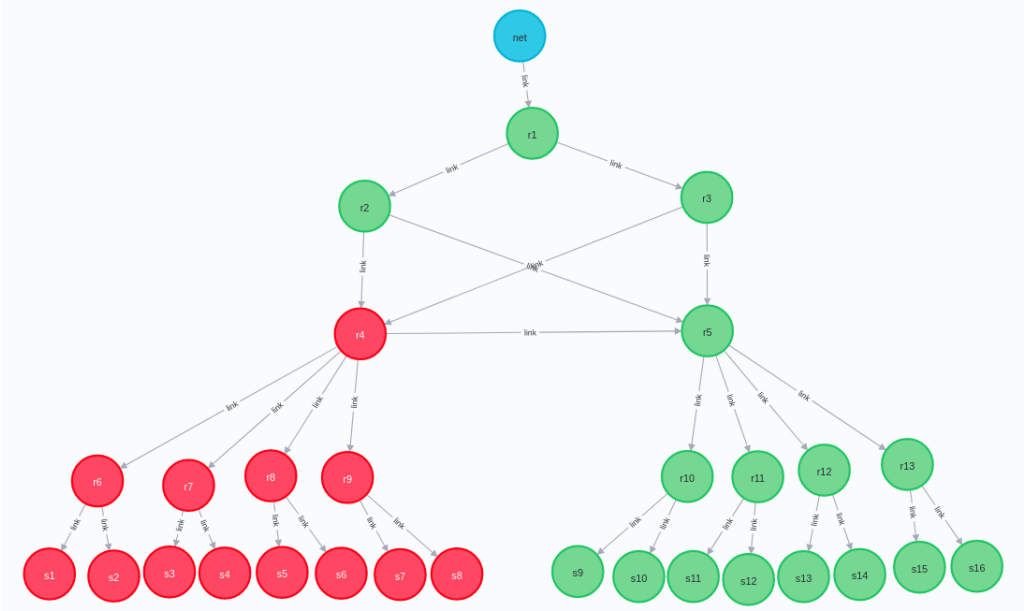

Figura 34: Comportamento da rede com base no Cenário 7

```
leo@leo-X510UR:~/Documents$ sudo python3 rootcause.py
Os equipamentos causa raiz da queda da rede:  ['r4 ']
Equipamentos secundarios:  ['r6 ', 'r7 ', 'r8 ', 'r9 ']
                           ['s1 ', 's2 ', 's3 ', 's5 ', 's6 ', 's7 ', 's8 ']
leo@leo-X510UR:~/Documents$
```

Figura 35: Resultado da RCA para o Cenário 7

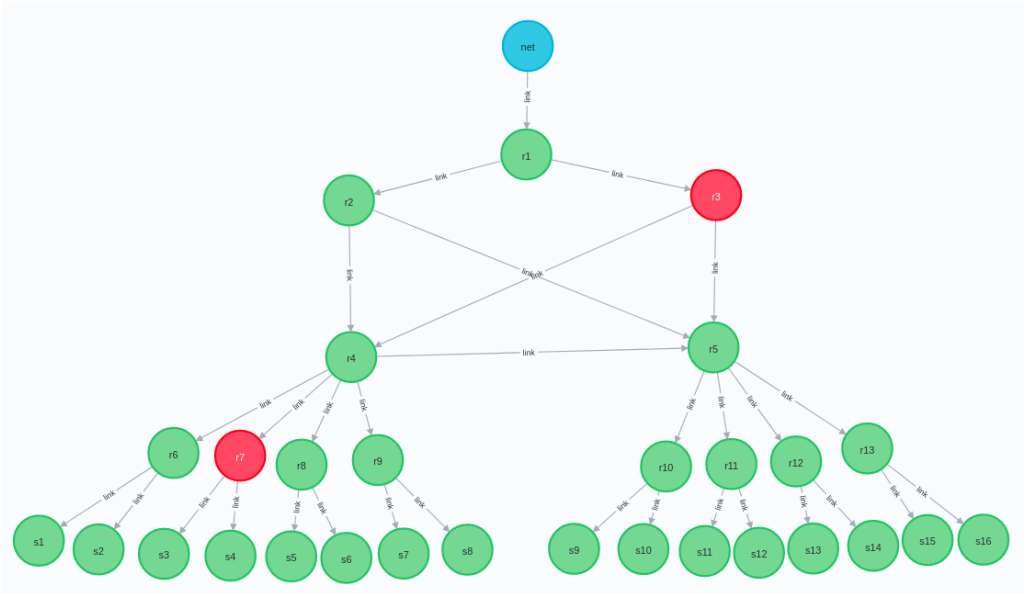

Figura 36: Equipamentos com falha do Cenário 8

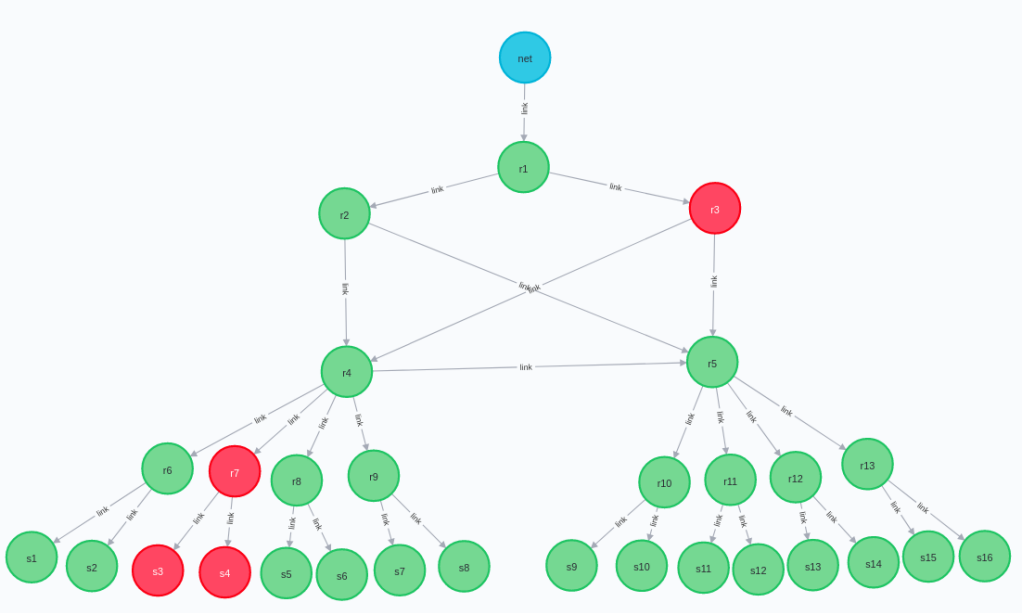

Figura 37: Comportamento da rede com base no Cenário 8

```
leo@leo-X510UR:~/Documents$ sudo python3 rootcause.py
Os equipamentos causa raiz da queda da rede:  ['r3 ', 'r7 ']
Equipamentos secundarios:  ['s3 ', 's4 ']
leo@leo-X510UR:~/Documents$
```

Figura 38: Resultado da RCA para o Cenário 8

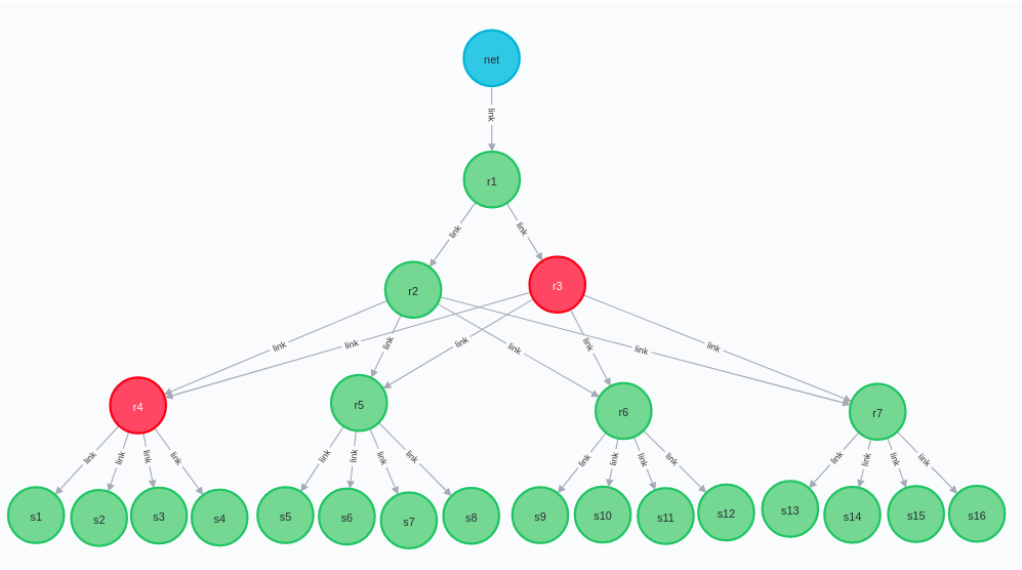

Figura 39: Equipamentos com falha do Cenário 9

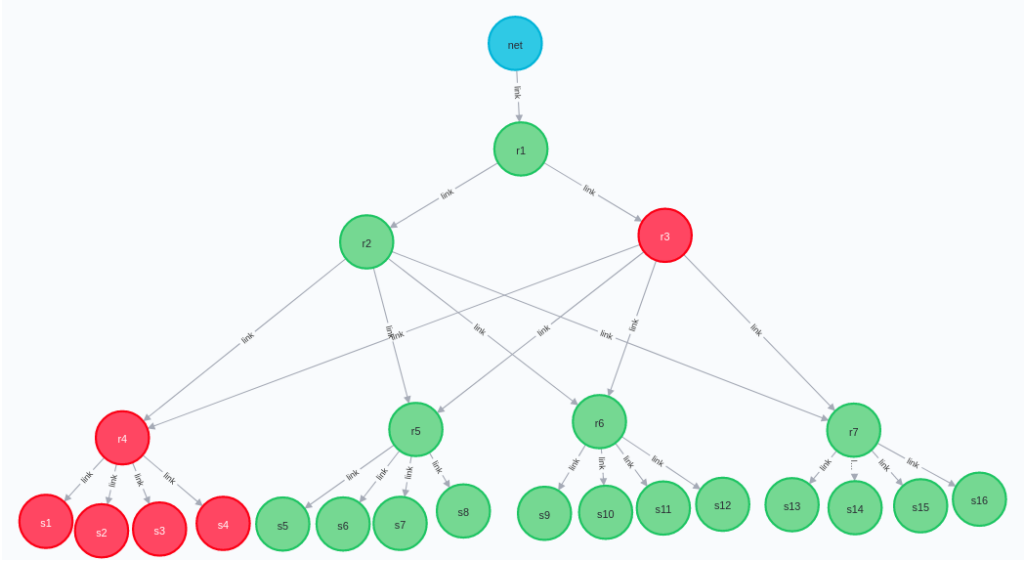

Figura 40: Comportamento da rede com base no Cenário 9

```
leo@leo-X510UR:~/Documents$ sudo python3 rootcause.py
Os equipamentos causa raiz da queda da rede:  ['r3 ', 'r4 ']
Equipamentos secundarios:  ['s1 ', 's2 ', 's3 ', 's4 ']
leo@leo-X510UR:~/Documents$
```

Figura 41: Resultado da RCA para o Cenário 9

Por fim, os cenários descritos na Tabela 4, não foram possíveis de serem simulados, uma vez que envolviam quedas de interfaces específicas e não de todas, como quando ocorre um equipamento tem perda total de conectividade. Isso ocorreu pois o hardware da máquina utilizada para simular as redes não era poderoso o suficiente para permitir que as redes, que possuíam mais de dez equipamentos, estivessem ativas e, ao mesmo tempo, realizar o acesso ao terminal de controle dos equipamentos a fim de introduzir falhas em certas interfaces. Entretanto, vale notar que, caso isso fosse possível, a estratégia para diagnosticar falhas em interfaces seria a mesma, porém acrescida de um passo a mais que consistiria em: ao obter os equipamentos sem conectividade *(status Disconnected)* cujos equipamentos "pais" possuem conectividade *(status Connected)* porém ao invés de apontar este equipamento como causa raiz, seria verificado se os demais equipamentos que possuem conexão os seus equipamentos "pai", ou seja: equipamentos "irmãos", possuem conectividade. Em caso afirmativo, então o equipamento com a queda da interface de comunicação entre o equipamento "pai" e o equipamento em questão seria apontada como causa raiz; porém, se nenhum dos "irmãos" possuísse conexão, então o equipamento em questão seria apontado como causa raiz.

6 Conclusão

Conforme foi possível observar pelos resultados, a análise de causa raiz foi capaz de identificar a causa, e apontar os equipamentos que poderiam ter seus problemas disfarçados em ordem decrescente de probabilidade, tanto em cenários nos quais apenas um equipamento apresentava falha quanto em cenários em que havia equipamentos com falha em mais de um equipamento da rede e estes não estavam diretamente conectados. Em cenários nos quais os equipamentos com falha estavam diretamente ligados, isto é: um deles era responsável por prover conexão para o outro, a análise apontou somente como causa raiz o equipamento "pai"; entretanto, o segundo equipamento causador da falha na rede foi apontado como o mais provável de possuir um problema disfarçado. Vale notar quapresentae a topologia *multi-tier* apresenta maior resiliência para quedas que ocorrem na camada *Core* ou *Aggregation* devido à sua natureza mas a topologia *Spine-Leaf* não a situação de erro observada em *multi-tier* e descrita acima.

Como dito anteriormente na seção de resultados, não foi possível simular a queda de apenas certas interfaces em um equipamento, o que impactaria a conectividade de apenas um e não de todos os equipamentos conectados, por falta de poder computacional. Entretanto, a estratégia para resolução deste cenário não difere consideravelmente da que foi implementada e poderia sê-la sem grandes modificações no *script*.

Por fim, temos aqui alguns pontos que poderiam gerar melhorias sensíveis nessa análise: além do uso de um hardware mais poderoso que permita emular equipamentos de telecomunicações que são de fato encontrados em redes corporativas, existem etapas do processo que foram feitas manualmente e que poderiam ser automatizadas como, por exemplo: a modelagem da rede simulada no GNS3 para o Neo4j. Além disso, o processo de análise causa raiz poderia ser disparado automaticamente quando um *threshold*, definido pela equipe de operação da rede, de alarmes de perda de conectividade fosse detectado. Também seria interessante expandir essa análise para casos de perda de pacotes em interfaces.

Referências

[1] A. VIJAY KUMAR; G. ANJAN BABU, *Netwok Failures and Root Cause Analysis: an approach using Graph Databases,* Department of Computer Science, S V University, Tirupati, India (2015).

[2] J. SCHOENFISCH; C. MEILICKE; J.V STÜLPNAGEL; J. ORTMANN; H STUCKENSCHMIDT,*Root cause analysis in IT infrastructures using ontologies and abduction in Markov Logic Networks,* ELSERVIER. Information Systems, p 103 - 116 (2017)

[3] CISCO, *Cisco Data Center Infrastructure 2.5 Design Guide,* Cápitulo 2 (2011)

[4] CISCO, *Cisco Data Center Spine-and-Leaf Architecture: Design Overview White Paper* (2020)

[5] GNS3, *https://www.gns3.com/*

[6] S. L. LOH ;C. K. GAN; T. H. CHEONG; S. SALLEH; N. H. SARMIN, *An Overview on Network Diagrams: Graph-Based Representation* (2015)

[7] NEO4J, *neo4j,* disponível em: https://neo4j.com/

[8] NEO4J, *Cypher Query Language,* disponível em: https://neo4j.com/developer/cypher/

[9] NEO4J, *The Shortest Path algorithm,* disponível em: https://neo4j.com/docs/graph-algorithms/current/labs-algorithms/shortest-path/