

Projektovanje i implementacija aplikacije za vizualizaciju topologije lokalnih računarskih mreža

Vukašin Dobromirović

Sadržaj – U ovom radu je prikazana aplikacija pomoću koje je moguće vizualizovati računarske mreže u cilju boljeg razumevanja.

I. UVOD

U današnjem digitalnom dobu, mrežna infrastruktura predstavlja ključnu stavku za funkcionisanje modernih organizacija i preduzeća. Upravljanje mrežom je postalo neophodno kako bi se osigurala stabilnost, bezbednost i efikasnost komunikacije između različitih uređaja i servera. U tom kontekstu, praćenje topologije mreže i identifikacija povezanih uređaja je od suštinskog značaja.

Izazov koji se nameće u ovoj oblasti je kako efikasno prikupiti informacije o uređajima na mreži, identifikovati njihove veze i konfiguraciju, te na kraju, prikazati ove podatke na način koji je pregledan i informativan. Tradicionalno, ovakvo prikupljanje informacija zahteva ručno interagovanje sa svakim uređajem, što može biti vremenski zahtevno i podložno greškama.

U cilju rešavanja ovog problema, ovaj rad predstaviće aplikaciju koja omogućava virtualizaciju topologije lokalnih računarskih mreža. Ovaj program integriše različite biblioteke i tehnologije kako bi efikasno obavio ovaj zadatak. Kroz analizu i prikazivanje topologije mreže, ovaj alat omogućava administratorima da bolje razumeju kako su uređaji povezani, identifikuju potencijalne probleme i brže reaguju na potrebe za održavanjem mreže.

II. PREGLED TEHNIKA NEOPHODNIH ZA IMPLEMENTACIJU APLIKACIJE

A. Cisco operativni sistem (Cisco IOS)

Cisco [1] operativni sistem, ili skraćeno Cisco IOS [2], je operativni sistem koji se koristi na Cisco mrežnim uređajima kao što su ruteri i svičevi. Ovaj operativni sistem je ključan za upravljanje mrežom i obezbeđuje korisnicima širok spektar funkcionalnosti za konfiguraciju i upravljanje uređajima. U implementaciji programa, Cisco OS je osnova za komunikaciju sa Cisco uređajima putem SSH protokola.

V. Dobromirović, student treće godine osnovnih studija na modulu Računarstvo i informatika Elektronskog fakulteta u Nišu, Aleksandra Medvedeva 14, 18000 Niš, Serbia, E-mail: vukasindobromirovic@elfak.rs

B. CDP komanda

CDP (*Cisco Discovery Protocol*) [3] je Cisco-ov protokol za otkrivanje susednih uređaja u mreži. Ovaj protokol omogućava uređajima da razmenjuju informacije o svojim identitetima, povezanim interfejsima i drugim relevantnim podacima. Aplikacija projektovana u ovom radu koristi CDP komandu “*show cdp neighbors detail*” kako bi prikupila informacije o susednim uređajima, interfejsima i IP adresama.

C. SSH protocol (Secure Shell)

SSH (*Secure Shell*) [4] je protokol za sigurnu komunikaciju i daljinsko upravljanje sa udaljenim uređajima u mrežnom okruženju. Osnovna svrha SSH protokola je omogućavanje bezbedne razmene podataka između klijenta (npr. računar i administrator) i udaljenih servera ili uređaja, kao što su svičevi, ruteri ili serveri. Prilikom povezivanja sa uređajima, SSH omogućava autentikaciju korisnika i uspostavljanje sigurne veze. To znači da se informacije kao što su korisnička imena, lozinke i prikupljeni podaci šifriraju kako bi se sprečilo prisluškivanje ili neovlašćeni pristup. Osim toga, SSH pruža mehanizme za autentikaciju i zaštitu integriteta podataka.

D. BFS algoritam (Breadth-First Search)

BFS (*Breadth-First Search*) [5] je algoritam za pretraživanje ili kretanje kroz stablo ili graf strukturu podataka. U kontekstu aplikacije, BFS se koristi za skeniranje mreže. Počevši od ciljnog uređaja (*gateway*) [6], BFS se širi na susedne uređaje i prikuplja informacije o njihovim povezanim uređajima i interfejsima. Ovaj algoritam omogućava temeljno istraživanje mreže i identifikaciju svih povezanih uređaja.

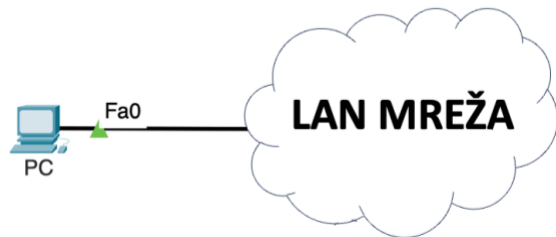
Pregled ovih ključnih tehnika i tehnologija omogućava bolje razumevanje implementacije i kako program efikasno prikuplja informacije o mreži. Ovaj pregled će takođe postaviti temelje za kasnije delove rada koji će se baviti detaljima implementacije i praktičnom primenom programa.

III. PROJEKTOVANJE APLIKACIJE

A. Princip ulazne tačke

Program je od značaja za sve mrežne administratore u cilju boljeg razumevanja i upravljanja mrežom. Veza se uspostavlja sa gejtvajom, koji predstavlja ulaznu tačku na

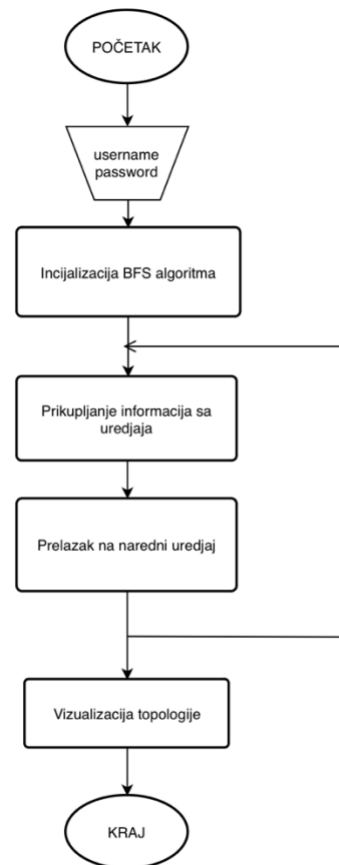
mrežu. Na slici 1 se može videti slikovit opis. Nakon što je veza sa gejtvejom uspostavljena, CDP (*Cisco Discovery Protocol*) naredba se koristi za postepeno otkrivanje uređaja i ostatka mreže. Na ovaj način, mrežna mapa se sistematski gradi putem programa, omogućavajući administratorima da steknu dublji uvid u topologiju mreže i identifikaciju uređaja i resursa koji čine deo lokalne mreže.



Slika 1. Princip povezivanja na ulaznoj tački.

B. Algoritam aplikacije

Na slici 2 je prikazan algoritam rada predložene aplikacije. Na početku izvršenja korisnik zadaje korisničko ime i šifru za ssh pristup uređajima. Podrazumeva se da administrator ima iste kredencijale za pristup svim uređajima. Nakon unosa potrebnih podataka poziva se funkcija *get_default_gateway* pomoću koje se saznaje ip adresa gejtveja. Inicijalizuje se BFS (*Breadth- First Search*) algoritam i za početni čvor se uzima adresa gejtveja. BFS algoritam se primenjuje na mrežu kako bi se sistematski otkrivali uređaji i njihove veze. Za svaki pronađeni uređaj se izvršava CDP komanda kako bi se saznale dodatne informacije o susedima i interfejsima. Informacije o svakom uređaju, uključujući IP adrese, interfejse i veze, prikupljaju se tokom izvršavanja BFS algoritma. Ove informacije se beleže i čuvaju za kasniju analizu. Nakon završetka skeniranja i prikupljanja informacija, razmatra se način vizualizacije topologije mreže. Odgovarajuće biblioteke za crtanje grafova ili druge tehnike koriste se za grafički prikaz rezultata. Algoritam se testira na različitim mrežama kako bi se osiguralo da ispravno funkcioniše i da zadovoljava očekivanja.



Slika 2. Algoritam aplikacije.

IV. KREIRANJE APLIKACIJE

U nastavku će biti predstavljene određene funkcije i biblioteke koje su ključne za aplikaciju.

A. Potrebne biblioteke kao pomoć u izradi aplikacije

Na početku biće upotrebljeni neki standardni paketi u programskom jeziku Python. Paramiko [7] je *python* biblioteka koja omogućava implementaciju SSH protokola. Netmiko [8] je *python* biblioteka koja se koristi za automatizaciju mrežnih uređaja. Omogućava programerima da se povežu sa različitim mrežnim uređajima. *Networkx* [9] je *Python* biblioteka za analizu i manipulaciju grafova i mreža. Koristi se za proučavanje i vizualizaciju odnosa između entiteta u mreži, kao što su uređaji ili čvorovi, i njihove veze ili grane. *Matplotlib* [10] je *Python* biblioteka za crtanje grafova i dijagrama. Koristi se za vizualizaciju podataka i rezultata, uključujući grafove mreža, histograme, dijagrame rasipanja i mnoge druge vrste grafičkih prikaza. Na slici 3 je prikazan isečak koda kojim se uključuju u aplikaciju već pomenute biblioteke.

```
import paramiko
from netmiko import ConnectHandler
import networkx as nx
import matplotlib.pyplot as plt
```

Slika 3. Potrebni paketi u programskom jeziku Python.

B. Funkcija za čitanje podrazumevane adrese uređaja sa koga se pokreće aplikacija

Funkcija `get_default_gateway` ima za cilj da pronađe podrazumevani gejtvej odnosno ip adresu rutera mreže na kojoj je povezan uređaj sa koga se pokreće aplikacija. Kod funkcije se može videti na slici 4.

```
def get_default_gateway():
    try:
        output = os.popen("route
        print").read()
        gateway_pattern = r"0\.0\.0\.
        0\s+0\.0\.0\.0\s+(\d+\.\d+\.\d
        +\.\d+)\s+"
        match = re.search
        (gateway_pattern, output)
        if match:
            return match.group(1)
    except Exception as e:
        print(f"Greška: {e}")
    return None
```

Slika 4. Funkcija `get_default_gateway`.

C. Funkcija za uspostavljanje veze sa uređajem i prikupljanje informacija o samom uređaju

Funkcija `gather_device_info` ima zadatak da prikupi informacije o udaljenom mrežnom uređaju putem SSH protokola. Funkcija koristi biblioteke `paramiko` i `netmiko` koje su objašnjene na početku poglavlja. Kod funkcije se može videti na slici 5.

```
def gather_device_info(device_ip, username, password,
enable_password):
    # Parametri za SSH konekciju
    ssh_params = {
        'device_type': 'cisco_ios',
        'ip': device_ip, 'username': username,
        'password': password, 'secret': enable_password
    }
    try:
        ssh_session = ConnectHandler(**ssh_params)
        output = ssh_session.send_command('show cdp
        neighbors detail')
        ssh_session.enable()
        shrun = ssh_session.send_command('show
        running-config')
        ssh_session.disconnect()

        return output, shrun
    except Exception as e:
        print(f"Greška pri povezivanju ili izvršavanju
        komande na uređaju {device_ip}: {str(e)}")
```

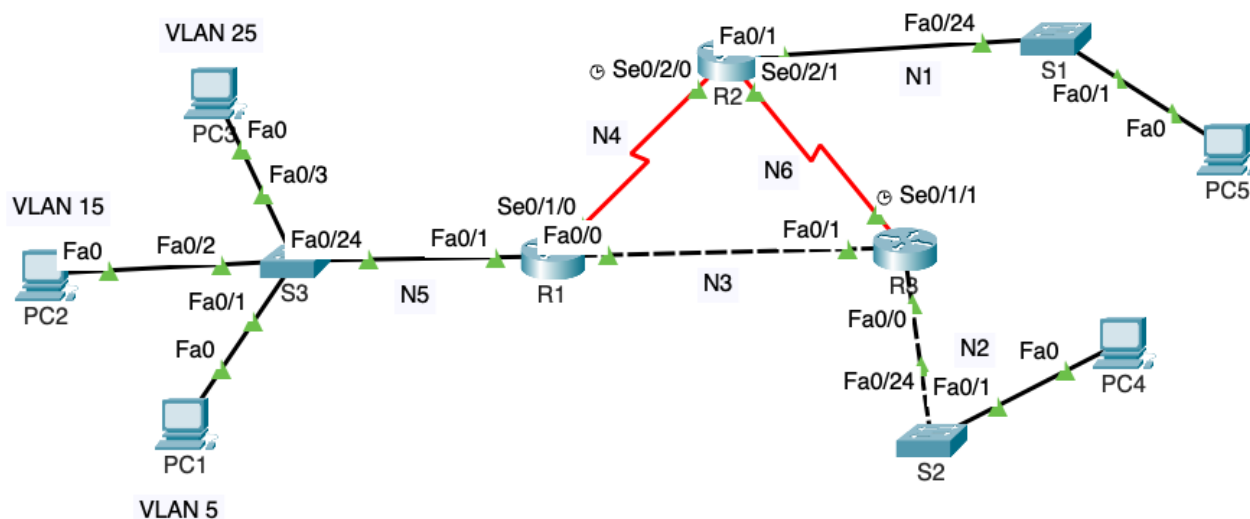
Slika 5. Funkcija `gather_device_info`.

D. Funkcija za iscrtavanje topologije

Funkcija `draw_custom_graph` se koristi za iscrtavanje grafa koji prikazuje mrežnu topologiju na osnovu podataka o čvorovima (uređajima) i granama (vezama) između njih. Ova funkcija koristi biblioteku `networkx` za manipulaciju grafovima i `matplotlib` za crtanje grafičkog prikaza. Detaljan kod možemo videti na slici 6.

```
def draw_custom_graph(nodes, edges, ip_addresses=None):
    G = nx.DiGraph()
    G.add_edges_from(edges)
    node_labels = {}
    for node in nodes:
        label = node
        if ip_addresses and node in ip_addresses:
            label += f'\n{ip_addresses[node]}'
        node_labels[node] = label
    edge_colours = ['black']
    pos = nx.spring_layout(G)
    node_font = {'fontname': 'Arial', 'size': 12}
    nx.draw_networkx_nodes(G, pos, cmap=plt.get_cmap
    ('jet'), node_color='lightblue', node_size=500)
    nx.draw_networkx_labels(G, pos, labels=node_labels,
    font_color='black', font_size=8, font_weight='bold',
    font_family=node_font
    ['fontname'])
    nx.draw_networkx_edges(G, pos, arrows=True,
    edge_color=edge_colours)
    plt.show()
```

Slika 6. Funkcija za iscrtavanje topologije.



Slika 9. Mreža nad kojom je izvršeno testiranje.

V. SLUČAJEVI UPOTREBE

Aplikacija za vizualizaciju topologije lokalnih računarskih mreža je realizovana i testirana na Elektronskom fakultetu u okviru Cisco akademije [11] u laboratoriji 527. Za kreiranje aplikacije korišćen je Python programski jezik i korišćeni su sledeći mrežni uređaji: Cisco 2811 ruteri i Cisco Catalyst 2960 svičevi, prikazani na slikama 7 i 8.



Slika 7. Cisco Catalyst 2960.



Slika 8. Cisco 2811.

A. Primer mreže na kojoj je izvršeno testiranje

Aplikacija će biti testirana na topologiji prikazanoj na slici 9. Uočavamo šest mreža označenim simbolima N1, N2 redom do N6. Svaka mreža ima određen opseg adresa. Za N1 je to 172.16.1.0/24, za N2 je 172.16.2.0/24, za N3 je 192.168.3.0/30, za N4 je 192.168.4.0/30, za N6 je 172.16.6.0/24. Što se tiče mreže N6 tu se koristi Router-on-stick [13] tehnika. Na mreži se mogu primetiti VLAN 5,

VLAN 15, VLAN 25, svaki redom u opsegu 10.0.5.0, 10.0.15.0 i 10.0.25.0 sa istom maskom /24.

B. Izvršenje aplikacije i analiza izlaznih podataka

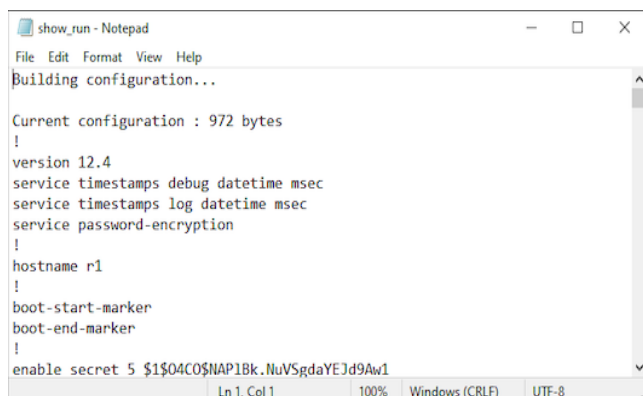
Nakon pokretanja aplikacije od korisnika se očekuje da unese podatke kao što su *username*, *password* i *password privileged mode* [12]. U međuvremenu, aplikacija pokreće prethodno opisanu funkciju *get_default_gateway* i saopštava ip adresu rutera odakle počinje skeniranje. U našem primeru je to ruter R1 pošto je aplikacija pokrenuta sa računara označenim PC1. Potrebno je zatim pritisnuti taster *Enter* i aplikacija pokreće skeniranje mreže.

Nakon završetka aplikacije u konzoli se ispisuju rezultati. Za svaki uređaj, bilo to da je svič ili ruter, dobijamo detaljne informacije kako je povezan sa svojim susedom, odnosno preko kog interfejsa. Konzolni izlaz aplikacije je prikazan na slici 10.

Uređaj čiji je hostname "r1" i ip adresa "10.0.5.1" je:
Povezan preko interfejsa "FastEthernet0/1" sa uređajem čiji je hostname "s3".
Povezan preko interfejsa "Serial0/1/0" sa uređajem čiji je hostname "r2".
Povezan preko interfejsa "FastEthernet0/0" sa uređajem čiji je hostname "r3".
Uređaj čiji je hostname "s3" i ip adresa "10.0.5.6" je:
Povezan preko interfejsa "FastEthernet0/24" sa uređajem čiji je hostname "r1".
Uređaj čiji je hostname "r2" i ip adresa "192.168.4.2" je:
Povezan preko interfejsa "Serial0/2/1" sa uređajem čiji je hostname "r3".
Povezan preko interfejsa "Serial0/2/0" sa uređajem čiji je hostname "r1".
Povezan preko interfejsa "FastEthernet0/1" sa uređajem čiji je hostname "S1".
Uređaj čiji je hostname "r3" i ip adresa "192.168.3.2" je:
Povezan preko interfejsa "Serial0/1/1" sa uređajem čiji je hostname "r2".
Povezan preko interfejsa "FastEthernet0/1" sa uređajem čiji je hostname "r1".
Povezan preko interfejsa "FastEthernet0/0" sa uređajem čiji je hostname "S2".
Uređaj čiji je hostname "S1" i ip adresa "172.16.1.100" je:
Povezan preko interfejsa "FastEthernet0/24" sa uređajem čiji je hostname "r2".
Uređaj čiji je hostname "S2" i ip adresa "172.16.2.100" je:
Povezan preko interfejsa "FastEthernet0/24" sa uređajem čiji je hostname "r3".

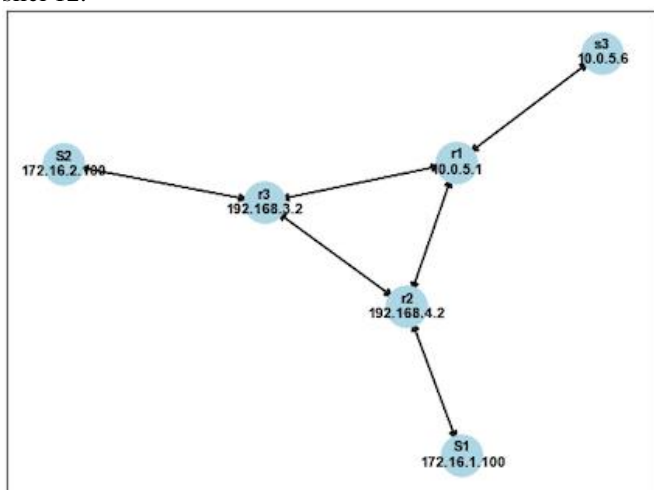
Slika 10. Konzolni prikaz nakon završetka aplikacije.

Takođe, može se uočiti kreirani fajl po nazivom *show_run.txt*. Fajl sadrži detaljni opis svakog uređaja na kome je bio izvršen skok. Na slici 11 je prikazan i sam fajl.



Slika 11. Fajl sa detaljnom konfiguracijom uređaja.

Treći izlazni podatak je vizualizacija mreže kreirane prethodno opisanom funkcijom *draw_custom_graph*. Uočava se povezanost koja je identična konzolnom prikazu. Topologija je prikazana na slici 12.



Slika 12. Vizualizacija mreže.

VI. ZAKLJUČAK

Mreže postaju sve kompleksnije što čini važnim razumevanje kako su uređaji međusobno povezani i kako se saobraćaj kreće kroz mrežu. Ova aplikacija pruža sredstva za brzo i efikasno otkrivanje uređaja u mreži, analizu njihovih povezanosti i prikazivanje tih informacija u obliku grafičkog prikaza. Time se olakšava dijagnostika problema, optimizacija mreže i bolje razumevanje njenih karakteristika. U budućnosti, aplikacija se može proširiti i dodatno unaprediti dodavanjem naprednih funkcionalnosti, kao što su analiza sigurnosti mreže, automatsko konfigurisanje uređaja ili integracija sa drugim alatima za

upravljanje mrežom. Ovaj projekat služi kao temelj za dalje istraživanje i razvoj u oblasti mrežnih tehnologija.

ZAHVALNICA

Autor se zahvaljuje prof. dr Vladimiru Ćiriću na mentorstvu i podršci i prof. dr Emiliji Živanović na pomoći pri pisanju rada. Takođe, autor se zahvaljuje i Cisco akademiji na korišćenju laboratorije 527.

LITERATURA

- [1] Cisco: Networking, Cloud, and Cybersecurity Solutions n.d. Poslednji pristup 27.10.2023. <https://www.cisco.com>
- [2] Cisco Networking Academy, Cisco. Poslednji pristup 27.10.2023. <https://lms.netacad.com>
- [3] Cisco n.d. *Understanding and configuring CDP*. Poslednji pristup 27.10.2023. <https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst4500/12-2/25ew/configuration/guide/conf/cdp.pdf>
- [4] Cisco Networking Academy, Cisco. Poslednji pristup 27.10.2023. <https://lms.netacad.com>
- [5] https://en.wikipedia.org/wiki/Breadth-first_search
- [6] Wikipedia: The free Encyclopedia n.d. Poslednji pristup 27.10.2023. https://en.wikipedia.org/wiki/Default_gateway
- [7] Paramiko n.d. Poslednji pristup 27.10.2023. <https://pypi.org/project/paramiko/>
- [8] Kirk Byers [2023]. netmiko: Kolekcija informacija korišćenih u projektu. Github. <https://github.com/ktbyers/netmiko>
- [9] NetworkX – Network Analysis in Python 2014. Poslednji pristup 27.10.2023. <https://networkx.org>.
- [10] Matplotlib: Visualisation with Python 2012. Poslednji pristup 27.10.2023. <https://matplotlib.org>.
- [11] Cisco networking academy n.d. Poslednji pristup 27.10.2023. <https://www.netacad.com>
- [12] Controlling Switch Access with Passwords and Privilege Levels, Cisco. Poslednji pristup 27.10.2023. <https://www.cisco.com/en/US/docs/>
- [13] Wikipedia: The free Encyclopedia n.d. Poslednji pristup 27.10.2023. https://en.wikipedia.org/wiki/Router_on_a_stick