Descrierea temei:

Proiectul presupune studiul soluției de interconectare pentru rețele private virtuale de tip Hub and Spoke folosind DMVPN. Implementarea soluției de tunelare va fi proiectată peste o rețea de tip "Internet" dezvoltată cu ajutorul protocolului MPLS, această rețea având rolul de furnizor de servicii. Obiectivul proiectului este construirea rețelei unei firme cu mai multe sucursale poziționate în diferite puncte geografice peste rețeaua furnizorului, folosind simulatorul de rețele EVE-NG și echipamente Cisco. Se vor analiza parametri de performanța, precum latența, scalabilitatea și rata de pierdere a pachetelor în momentul congestiei, implementand tehnici de QoS (shaping și policing), aceste metrici urmând a fi extrase din echipamentele virtuale folosind Wireshark și prezentate cu ajutorul unei aplicații web construită în limbajul de programare Python (Django/Flask) și React JS/Angular/Vue/Bootstrap.

1. Introducere

Nevoia de comunicare intre persoane aflate la distanta a fost satisfacuta odata cu dezvoltarea „Internetului". Dar aceasta dezvoltare a adus odata cu ea mai multe pericole si amenintari. De aceea, in viata de zi cu zi se pune accent pe transmiterea informatiilor peste „Internet" intr-un mod sigur, securizat. Scopul securizarii transmisiunii este de a minimiza interceptarea informatiilor de catre atacatori.

Motivatie

Motivatia acestui proiect este data de contextul actual al situatiei post-pandemice din punct de vedere al companiilor cu angajati ce lucreaza de acasa. Pentru aceste entitati juridice, securitatea este un laitmotiv si influenteaza dezvoltarea lor in domeniul de activitate. Din aceasta cauza, majoritatea companiilor apeleaza la furnizorii de servicii de retea. In aceasta teza se prezinta un tip de serviciu oferit de furnizori, reteaua privata virtuala, implementand tehnologia DMVPN.

Aplicabilitate

Proiectul vizeaza in special persoanele juridice cu mai mult de 2 sedii aflate la distante mari, acestea fiind cele ce contacteaza adesea furnizorii de servicii. Acest proiect va implementa un exemplu bazat pe o companie cu 3 sucursale si un sediun principal, care vor comunica intre ele printr-o retea privata virtuala dezvoltata deasupra retelei furnizorului de servicii, aceasta retea fiind de tipul „Internet".

Obiective

Obiectivele aceste lucrari sunt:

- Realizarea unei retele private virtuale peste Internet
- Analizarea performantelor retelei cu ajutorul unei interfete web
- Automatizarea unor procese in caz de congestie

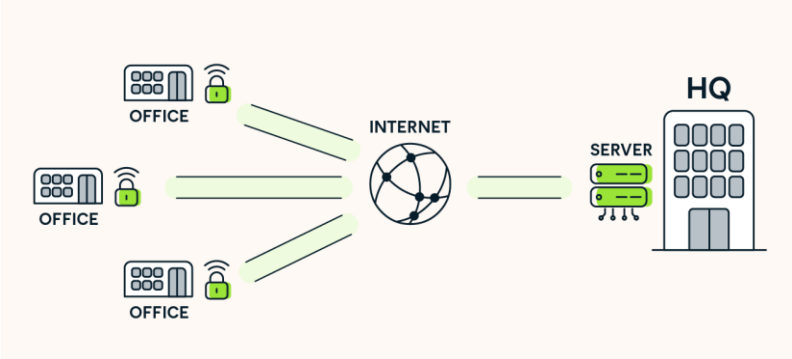
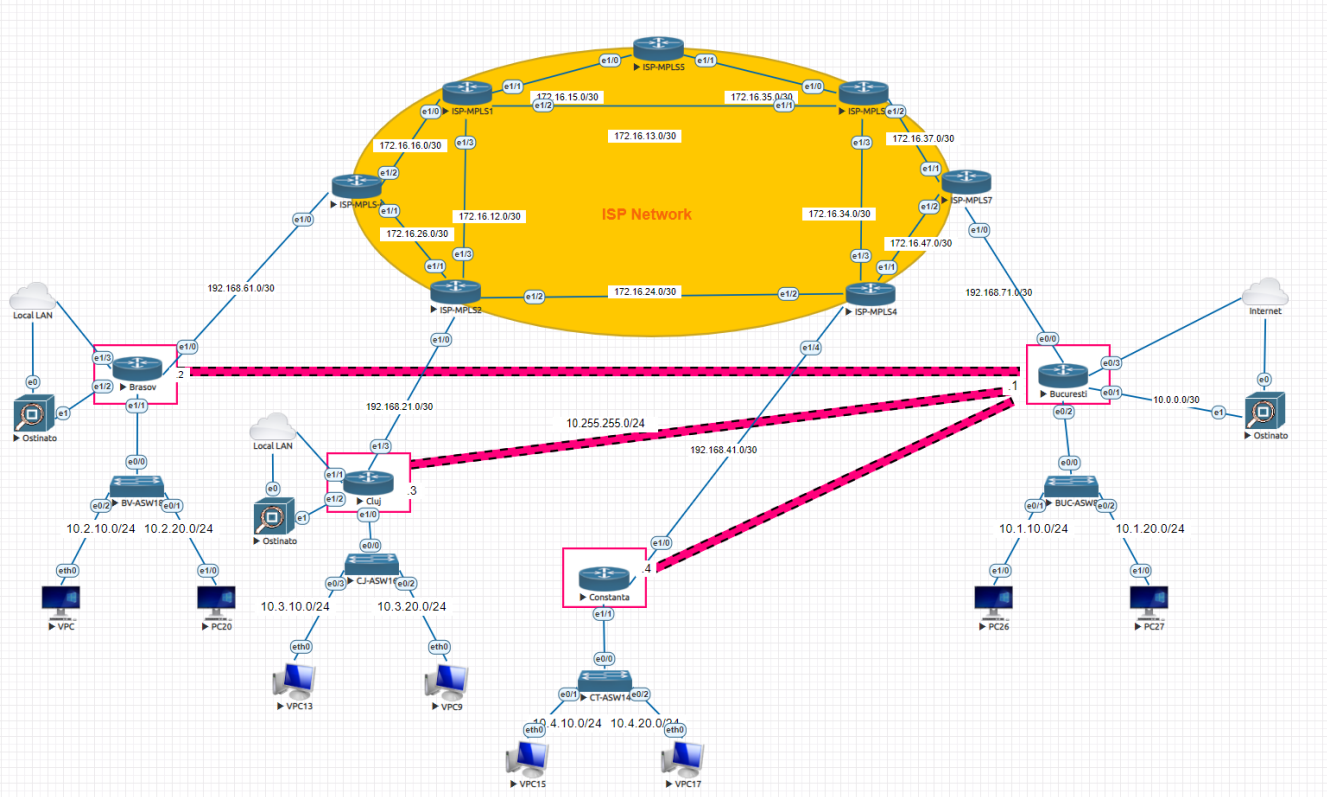


Fig. 1

Sarcini finalizate pana la acest moment:

Stabilirea dimensiunii retelei, a numarul de routere si switch-uri in functie de capacitatea PC-ului care ruleaza serverul de EVE-NG. Astfel, s-a ajuns la topologia de mai jos:



Planul de adresare pentru toata reteaua este descris in tabelul de mai jos:

ISP Network	ISP-MPLS1	ISP-MPLS2	ISP-MPLS3	ISP-MPLS4	ISP-MPLS5	ISP-MPLS6	ISP-MPLS7
ISP-MPLS1	N/A	172.16.12.2/30 (e1/3)	172.16.13.2/30 (e1/1)	N/A	172.16.15.2/30 (e1/0)	172.16.16.2/30 (e1/2)	N/A
ISP-MPLS2	172.16.12.1/30 (e1/3)	N/A	N/A	172.16.24.2/30 (e1/2)	N/A	172.16.26.2/30 (e1/1)	N/A
ISP-MPLS3	172.16.13.1/30 (e1/2)	N/A	N/A	172.16.34.2/30 (e1/3)	172.16.35.2/30 (e1/1)	N/A	172.16.37.2/30 (e1/1)
ISP-MPLS4	N/A	172.16.24.1/30 (e1/2)	172.16.34.1/30 (e1/3)	N/A	N/A	N/A	172.16.47.2/30 (e1/2)
ISP-MPLS5	172.16.15.1/30 (e1/1)	N/A	172.16.35.1/30 (e1/0)	N/A	N/A	N/A	N/A
ISP-MPLS6	172.16.16.1/30 (e1/0)	172.16.26.1/30 (e1/1)	N/A	N/A	N/A	N/A	N/A
ISP-MPLS7	N/A	N/A	172.16.37.1/30 (e1/2)	172.16.47.1/30 (e1/1)	N/A	N/A	N/A

Link Network Between DMVPN and ISP routers: .1 for ISP, .2 for DMVPN				
	ISP-MPLS7 (e1/0)	ISP-MPLS4 (e1/4)	ISP-MPLS2 (e1/0)	ISP-MPLS6 (e1/0)
Bucuresti (e0/0)	192.168.71.0/30	N/A	N/A	N/A
Constanta (e1/0)	N/A	192.168.41.0/30	N/A	N/A
Cluj (e1/3)	N/A	N/A	192.168.21.0/30	N/A
Brasov (e1/0)	N/A	N/A	N/A	192.168.61.0/30

DMVPN Overlay Network Tunnel Addreses: 10.255.255.0/24	
Bucuresti (Tunnel0)	10.255.255.1
Brasov (Tunnel0)	10.255.255.2
Cluj (Tunnel0)	10.255.255.3
Constanta (Tunnel0)	10.255.255.4

LAN Network: 10.0.0.0/8		VLAN IT	VLAN Marketing
Bucuresti	10.1.0.0/16	10.1.10.0/24	10.1.20.0/24
Brasov	10.2.0.0/16	10.2.10.0/24	10.2.20.0/24
Cluj	10.3.0.0/16	10.3.10.0/24	10.3.20.0/24
Constanta	10.4.0.0/16	10.4.10.0/24	10.4.20.0/24
*default-gateway will always be .1 for every vlan			
**addressing will be done according to the interface from the scheme			

Se poate observa reteaua furnizorului de servicii incadrata intr-o forma rotunjita de culoare portocalie. S-a ales acest tip de conectare denumit Partial Mesh pentru a oferi caracteristica de redundanta specifica unei retele de tipul „Internet”. Astfel, se va testa mai departe raspunsul retelei noastre la defectul unui segment de retea din cadrul retelei furnizorului.

Mai departe, am propus urmatorul scenariu: suntem administratorii unei retele de dimensiuni medii, care are sediul principal in Bucuresti si alte 3 sedii cu angajati in Brasov, Cluj si Constanta. Ne dorim interconectarea tuturor acestor sedii, astfel incat sa fie posibila comunicarea intre ele. Astfel, vom crea o retea virtuala privata, peste reteaua furnizorului de servicii pentru a putea crea propriile reguli de comunicare in aceasta retea. Am ales sa cream o retea privata virtuala peste un ISP pentru a simula cat mai bine realitatea, deoarece este mult mai ieftin sa folosesti deja o infrastructura instalata pentru a conecta sucursale din orase diferite, decat sa iti creezi propria infrastructura cu propriile cabluri intre

orasele de interes. In plus, se rezolva si problema scalabilitatii: in cazul in care apare o sucursala noua, aceasta este mai usor de integrat in reteaua privata virtuala, decat de cablat la o infrastructura fizica creata de noi.

Tehnologia pe care am ales sa o folosesc pentru a dezvolta aceasta retea privata virtuala se numeste DMVPN (Dynamic Multipoint Virtual Private Network). Este o solutie care se preteaza cel mai bine pe scenariul propus, in care arhitectura retelei este de tipul Hub & Spoke, hub-ul la noi fiind sediul principal (Bucuresti) si spoke-urile sunt sucursalele din restul oraselor. DMVPN are la baza tuneluri GRE de tipul point-to-multipoint, precum si tunele GRE over IPSec, pentru a furniza securitate prin criptarea si integritatea informatiei transmise.

Pentru a configura reteaua furnizorului, s-a ales sa se foloseasca OSPF ca protocol de rutare intern, datorita usurintei cu care se configureaza acesta. Astfel, fiecare router al ISP-ului va anunta intr-un proces de ospf reteaua 172.16.0.0/16 cu comanda network, astfel creandu-se adiacente pe fiecare interfata care are IP-ul din aceasta retea.

router ospf 10
network 172.16.0.0 0.0.255.255 area 0
!

Mai mult decat atat, se doreste ca reteaua furnizorului sa fie eficienta, avand in vedere numarul mare de prefixe care se vor anunta prin aceasta retea si posibilitatea ca adresele IPv4 sa fie duplicate, router-ele din cadrul retelei ISP nu trebuie sa faca redirectionarea pachetelor pe baza IP. De aceea se va implementa MPLS (Multi Protocol Label Switching). In terminologia MPLS, router-ele din interiorul retelei ISP poarta numele de router P (Provider) iar router-ele de la marginea retelei ISP se numesc router PE (Provider Edge). Pentru a configura MPLS pe echipementele din interiorul retelei ISP s-a dat comanda mpls ip la nivelul fiecare interfete care face parte din aceasta retea:

ISP-MPLS6:
interface Ethernet1/1
ip address 172.16.26.2 255.255.255.252
mpls ip
!

Datorita faptului ca furnizorul de servicii are mai multi clienti, ale caror adrese IP pot fi duplicate, intrucat fiecare client va folosi o gama de adrese IPv4 din spatiul de adresare privat, se implementeaza in reteaua ISP tehnologia L3VPN, bazata pe separarea router-elor pe VRF-uri, in functie de fiecare client. Astfel, pentru reteaua noastra LAN vom avea VRF-ul numit IP_LAN_DE, creat astfel:

!
vrf definition IP_LAN_DE
rd 3209:1
route-target export 1:1
route-target import 1:1
!

Cu ajutorul acestui VRF, prefixele fiecarui client nu vor fi doar simple prefixe IPv4, vor purta numele de prefixe vpnv4, adica adresa IPv4 + RD (Route Distinguisher). Astfel, se poate face deosebirea intre 2 adrese IP echivalente (de ex: 192.168.0.100 al clientului A cu 192.168.0.100 al clientului B), intrucat pentru fiecare client exista un RD separat. Parametrul route-target este adaugat prefixului atunci cand se anunta prin retea pentru a se cunoaste din ce VRF a fost exportat si in ce VRF trebuie importat. Anuntarea acestor prefixe este realizata cu ajutorul protocolului MP-BGP, astfel se realizeaza adiacente de iBGP intre PE-urile furnizorului de servicii si adiacente de eBGP intre PE-CPE (unde CPE = router-ul de core al fiecarei locatii); adiacentele sunt facute pe IP-urile de loopback, pentru a evita pierderii adiacentei in momentul in care legatura de date pica. Pentru o configurare uniforma, este configurat un Route-Reflector in procesul de iBGP, astfel PE-urile vor crea adiacenta doar cu RR-ul aferent.

ISP-MPLS6:

```
router bgp 3209
 bgp log-neighbor-changes
 no bgp default ipv4-unicast
 neighbor 172.16.5.5 remote-as 3209
 neighbor 172.16.5.5 update-source Loopback0
 !
 address-family ipv4
 exit-address-family
 !
 address-family vpnv4
  neighbor 172.16.5.5 activate
  neighbor 172.16.5.5 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf IP_LAN_DE
  neighbor 192.168.61.2 remote-as 64700
  neighbor 192.168.61.2 activate
 exit-address-family
!
```

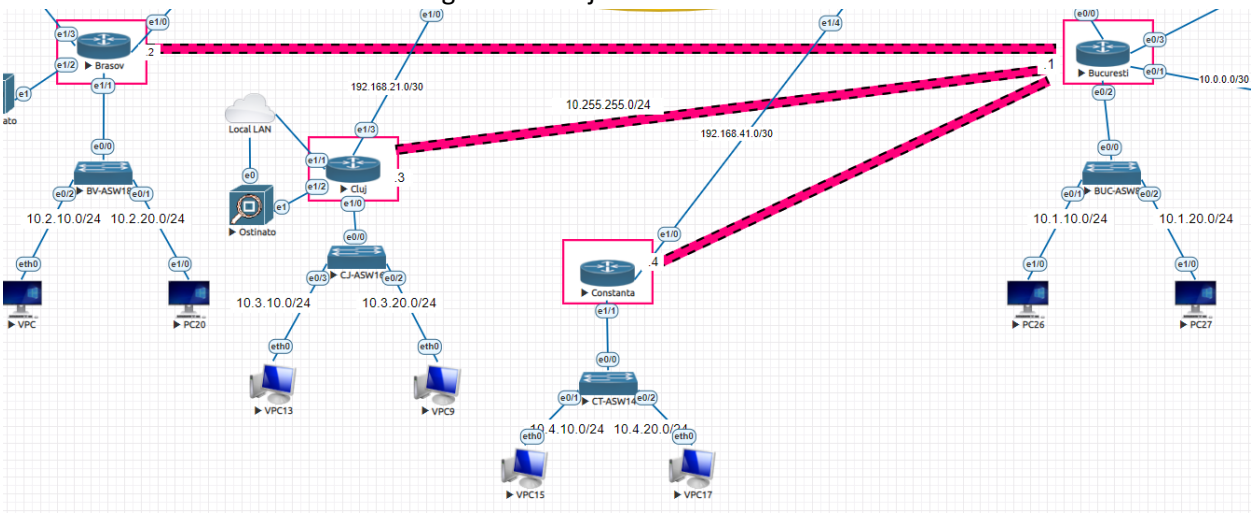
Se observa inchiderea address-family ipv4 si utilizarea in schimb address-family vpnv4. Route-Relfector-ul s-a creat cu urmatoarele comenzi:

ISP-MPLS5:

```
router bgp 3209
 bgp log-neighbor-changes
 no bgp default ipv4-unicast
 neighbor 172.16.2.2 remote-as 3209
 neighbor 172.16.2.2 update-source Loopback0
 neighbor 172.16.4.4 remote-as 3209
 neighbor 172.16.4.4 update-source Loopback0
 neighbor 172.16.6.6 remote-as 3209
```

```
 neighbor 172.16.6.6 update-source Loopback0
 neighbor 172.16.7.7 remote-as 3209
 neighbor 172.16.7.7 update-source Loopback0
 !
 address-family ipv4
 exit-address-family
 !
 address-family vpnv4
  neighbor 172.16.2.2 activate
  neighbor 172.16.2.2 send-community extended
  neighbor 172.16.2.2 route-reflector-client
  neighbor 172.16.4.4 activate
  neighbor 172.16.4.4 send-community extended
  neighbor 172.16.4.4 route-reflector-client
  neighbor 172.16.6.6 activate
  neighbor 172.16.6.6 send-community extended
  neighbor 172.16.6.6 route-reflector-client
  neighbor 172.16.7.7 activate
  neighbor 172.16.7.7 send-community extended
  neighbor 172.16.7.7 route-reflector-client
 exit-address-family
!
```

Reteaua interna este simbolizata in figura de mai jos:



DMVPN a fost configurat peste reteaua ISP-ului cu ajutorul protocolului NHRP. Configuratia Hub-ului (Bucuresti) este:

Bucuresti:

```
interface Tunnel0
 no shutdown
 ip address 10.255.255.1 255.255.255.0
 no ip redirects
 ip mtu 1400
 ip nhrp authentication donttell
 ip nhrp map multicast dynamic
 ip nhrp network-id 99
 ip nhrp holdtime 300
 ip tcp adjust-mss 1360
 ip ospf network point-to-multipoint
 ip ospf hello-interval 10
 delay 1000
 tunnel source 192.168.71.2
 tunnel mode gre multipoint
 tunnel key 100000
!
```

Iar configuratia unui Spoke (Brasov/Cluj/Constanta):

Brasov:

```
interface Tunnel0
 bandwidth 1000
 ip address 10.255.255.2 255.255.255.0
 no ip redirects
 ip mtu 1400
 ip nhrp authentication donttell
 ip nhrp map 10.255.255.1 192.168.71.2
 ip nhrp map multicast 192.168.71.2
 ip nhrp network-id 99
 ip nhrp holdtime 300
 ip nhrp nhs 10.255.255.1
 ip tcp adjust-mss 1360
 ip ospf network point-to-point
 delay 1000
 tunnel source 192.168.61.2
 tunnel mode gre multipoint
 tunnel key 100000
!
```

Astfel, s-a creat o retea privata virtuala unde se pot impune propriile filtre si propriile reguli de comunicare intre echipamente, fara a mai fi necesara o configuratie in reteaua furnizorului de servicii.

Mai departe, reteaua interna foloseste protocolul de rutare intern OSPF pentru a se anunta toate prefixele in cadrul aceluiasi proces:

Bucuresti:

```
router ospf 200
 network 7.7.7.7 0.0.0.0 area 0
 network 10.0.0.0 0.0.0.3 area 0
 network 10.1.0.0 0.0.255.255 area 0
 network 10.255.255.0 0.0.0.255 area 0
!
```

In acest moment, exista comunicatie end-to-end intre toate sucursalele, iar traficul este tunelat peste reteaua ISP-ului, cu ajutorul tehnologiei DMVPN.

Aspecte ce tin de reteaua LAN:

Pentru Bucuresti am ales ca router-ul principal sa aiba rol de server DHCP pentru intreaga retea, iar in fiecare site sa existe 2 VLAN-uri asociate cate unui rol din companie, de exemplu un vlan pentru echipa de IT si un vlan pentru echipa de Marketing.
Pentru server-ul DHCP au fost creata urmoarea configuratie:

```
!
ip dhcp excluded-address 10.4.10.1
ip dhcp excluded-address 10.4.20.1
ip dhcp excluded-address 10.3.10.1
ip dhcp excluded-address 10.3.20.1
ip dhcp excluded-address 10.2.20.1
ip dhcp excluded-address 10.2.10.1
ip dhcp excluded-address 10.1.10.1
ip dhcp excluded-address 10.1.20.1
!
ip dhcp pool CT10
 network 10.4.10.0 255.255.255.0
 default-router 10.4.10.1
!
ip dhcp pool CT20
 network 10.4.20.0 255.255.255.0
 default-router 10.4.20.1
!
ip dhcp pool CJ10
 network 10.3.10.0 255.255.255.0
 default-router 10.3.10.1
!
ip dhcp pool CJ20
 network 10.3.20.0 255.255.255.0
```

```
 default-router 10.3.20.1
!
ip dhcp pool BV10
 network 10.2.10.0 255.255.255.0
 default-router 10.2.10.1
!
ip dhcp pool BV20
 network 10.2.20.0 255.255.255.0
 default-router 10.2.20.1
!
ip dhcp pool BUC10
 network 10.1.10.0 255.255.255.0
 default-router 10.1.10.1
!
ip dhcp pool BUC20
 network 10.1.20.0 255.255.255.0
 default-router 10.1.20.1
!
```

Se observa ca intai s-au exclus adresele IP ale ruterelor aferente fiecarei locatii si s-a creat cate un pool pentru fiecare VLAN din fiecare locatie, astfel adresele IP ale tuturor statiilor din locatiile Bucuresti, Brasov, Cluj si constanta sunt obtinute prin DHCP.

```
BUC-PC26#sh ip int brief
Interface              IP-Address      OK? Method Status                Proto
FastEthernet0/0        unassigned      YES NVRAM  administratively down down
Ethernet1/0            10.1.10.2       YES DHCP   up                    up
```

De asemenea, am configurat si Inter-VLAN Routing cu Router-on-a-stick, ceea ce inseamna ca toate VLAN-urile pot comunica in acest moment intre ele. Dar acest lucru se poate modifica foarte usor, aplicand filtre la nivelul protocolului de rutare OSPF.

ROAS se configureaza pe sub-interfetele ruterului, specifice fiecarui VLAN, fiecare jucand rolul de default-gateway pentru vlan-ul corespunzator.

```
interface Ethernet0/2
 no shutdown
 no ip address
!
interface Ethernet0/2.10
 no shutdown
 encapsulation dot1Q 10
 ip address 10.1.10.1 255.255.255.0
!
```

interface Ethernet0/2.20
no shutdown
encapsulation dot1Q 20
ip address 10.1.20.1 255.255.255.0
!

Mai departe am inceput sa lucrez la interfata Web care se va ocupa sa extraga parametri de retea.

Folosesc framework-ul Django pentru partea de backend.

In acest moment, aceasta este pagina in care sunt preluati anumiti parametri de retea:

Trebuie mentionat ca este o pagina ce se reimprospateaza odata la un anumit interval de timp (am setat 5 secunde), astfel incat se realizeaza un HTTP GET de fiecare data cand se reimprospateaza. In acest fel, parametri sunt obtinuti in „real-time”.

Codul pentru obtinerea acestor parametri este scris intr-un view din Django, care arata asa:

```
def calcul_bw_bv(request):
    t1 = ReadCaptureBrasov()
    t1.start()

    start_time = None
    end_time = None
    bytes_total = 0
    protocols = {}
    packets_processed = 0

    for packet in t1.cap:
        if not start_time:
            start_time = float(packet.sniff_timestamp)
        end_time = float(packet.sniff_timestamp)
        bytes_total += int(packet.length)

        protocol = packet.highest_layer
        if protocol in protocols:
            protocols[protocol] += 1
        else:
            protocols[protocol] = 1

        packets_processed += 1
        if packets_processed == 1000:
            t1.cap.clear()
            packets_processed = 0

    elapsed_time = end_time - start_time
    bandwidth = bytes_total / elapsed_time
    # rezultatul in Bytes/s

    bandwidth = bandwidth * 8 / 1000
```

```
    # rezultatul in Kbps

    bw_list.append(bandwidth)

    params = []
    params.append(bw_list)
    params.append(protocols)

    context = {'params': params}

    return render(request, 'dmvpn/brasov_bw.html', context)
```

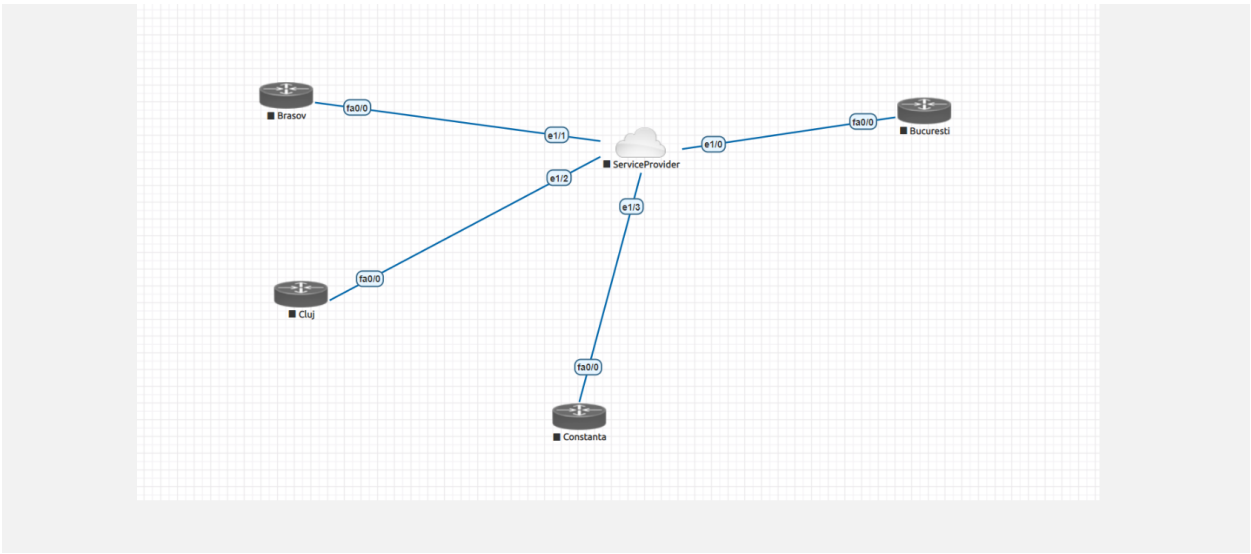
Captura de pachete este realizata cu ajutorul a 2 thread-uri, unul pentru pornire si unul pentru citire. In ambele thread-uri sunt folosite metode din libraria PyShark.

```
class StartCaptureBrasov(threading.Thread):

    def __init__(self):
        threading.Thread.__init__(self)
        self.stop_event = threading.Event()
        self.process = None

    def run(self):
        try:
            myBat =
open(r'C:\Users\Guci\Desktop\LicentaDjango\files\brasov.bat', 'w+')
            myBat.write('"C:\Program Files\EVE-NG\plink.exe" -ssh -batch -pw
eve root@192.168.0.99 "tcpdump -U -i '
                        'vunl0_10_16 -s 0 -w -" | "C:\Program
Files\Wireshark\Wireshark.exe" -k -i - -w
C:\\Users\\Guci\\Desktop\\LicentaDjango\\files\\brasov.pcap')
            myBat.close()

            self.process =
subprocess.Popen([r'C:\Users\Guci\Desktop\LicentaDjango\files\brasov.bat'])
            self.process.communicate()
        except Exception as e:
            print(e)

    def stop(self):
        if self.process:
            for proc in psutil.process_iter():
                if proc.name() == "Wireshark.exe" or proc.name() ==
"Dumpcap.exe":
                    proc.terminate()
            self.process.terminate()
        self.stop_event.set()


class ReadCaptureBrasov(threading.Thread):
    cap =
pyshark.FileCapture('C:\\Users\\Guci\\Desktop\\LicentaDjango\\files\\brasov.p
cap')
```

```
    def __init__(self):
        threading.Thread.__init__(self)
```

Thread-ul StartCapture este pornit la apasarea butonului „Afiseaza parametri”, din pagina urmatoare:

Pagina initiala arata in felul urmator:



Este o imagine ce permite cand dam click pe un router sa mergem pe pagina in care vor fi afisati parametri specifici fiecarui site.